Introduction to Secure Computation

Ronald Cramer

Dept. of Comp. Sc., ETH Zurich.
E-mail: `cramer@inf.ethz.ch`
URL: `http://www.inf.ethz.ch/personal/cramer`

Abstract. The objective of this paper[1] is to give an elementary introduction to fundamental concepts, techniques and results of Secure Computation.
Topics covered include classical results for general secure computation by Yao, Goldreich & Micali & Wigderson, Kilian, Ben-Or & Goldwasser & Wigderson, and Chaum & Crépeau & Damgaard.
We also introduce such concepts as oblivious transfer, security against malicious attacks and verifiable secret sharing, and for some of these important primitives we discuss realization.
This paper is organized as follows.
Part I deals with oblivious transfer and secure (general) two-party computation.
Part II discusses secure general multi-party computation and verifiable secret sharing.
Part III addresses information theoretic security and presents detailed but elementary explanations of some recent results in Verifiable Secret Sharing and Multi-Party Computation.
The importance of theory and general techniques often lies in the fact that the true nature of security is uncovered and that this henceforth enables to explore what is "possible at all". This then motivates the search for concrete and often specialized realizations that are more efficient. Nevertheless, many principles developed as part of the general theory are fundamental to the design of practical solutions as well.

[1] This paper is based on a lecture given by the author at the 1998 Aarhus Summerschool in Cryptography and Data Security. It first appeared in [28]. The current paper is a revision (*January 2000*): changes throughout Part III to enhance clarity and to correct errors, Appendix A has been added, and some additional references are given.

Table of Contents

III Information Theoretic Security

Part I

Secure Two-Party Computation

1 Oblivious Transfer and Match-Making

Suppose there are two politicians who want to find out whether they both agree on a certain matter. For instance, they may be discussing a controversial law that has been proposed. Clearly, they could decide just to announce to each other their opinion, and both of them could determine whether there is agreement. But this has a drawback that careful politicians may wish to avoid. If only one of them supports that controversial law, he may lose face.

In other words, what they need is a method allowing two players to figure out if they both agree but in such a way that if they don't, then any player that has rejected the matter has no clue about the other player's opinion. Moreover, they may want to be able to carry out the method over a distance.

Technically, we can model the situation as follows. There are two players, A and B, and each of them has a secret bit. Say that A has the bit a and B has the bit b.

They want to compute $a \cdot b$ (which corresponds to the logical AND of a and b, and hence it is 1 if and only if $a = b = 1$) so that

- *Correctness*: none of the players is led to accept a false result.
- *Fairness*: each learns the result $a \cdot b$.
- *Privacy*: each learns nothing more than what is implied by the result and the own input.

Indeed, if A for example holds $a = 0$, then $a \cdot b = 0$, regardless of the value of b. Therefore, B's choice b remains unknown to A in this case.

We construct a solution to this "Match-Making" problem based on an important primitive *Oblivious Transfer* (OT) (more precisely: "chosen one-out-of-two oblivious transfer"). An OT is a protocol between two players, a sender S and a receiver R, that achieves the following. S has two secret *input bits*, b_0 and b_1, and R has a secret *selection bit* s. At the end of the protocol, which may consist of a number of exchanges of information between S and R, R obtains the bit b_s, but without having obtained any information about the other bit b_{1-s} (sender security). On the other hand, S does not get any information about the selection bit s (receiver security).

We use $\mathrm{OT}(b_0, b_1, s) = b_s$ to denote the output of an OT protocol as a function of the inputs. It is useful to observe that b_s is actually equal to $(1 \oplus s)b_0 \oplus sb_1$, where the operations are the usual multiplication and addition of bits.

<u>Sender</u>	<u>Receiver</u>
In: $b_0, b_1 \in \{0,1\}$	**In**: $s \in \{0,1\}$

$$\xleftrightarrow{\quad S \to R : \mathrm{OT}(b_0, b_1, s) \quad}$$

	Out: b_s

Although at this point it is not clear whether OT-protocols exist and if so, how to construct them, we can already solve the Match-Making problem by assuming an OT-protocol!

This is how. If A and B now execute the OT-protocol with A acting as the sender and B as the receiver, using $b_0 = 0$ and $b_1 = a$, and $s = b$ as their respective inputs, we see that B gets the value ab as output. In other words, $\mathrm{OT}(0, a, b) = (b \oplus 1)0 \oplus ba = ab$. Finally, B simply reveals ab to A, so that both learn the result. And indeed, if $b = 0$, then $ab = 0$ no matter what a is and player B learns nothing more about a by the properties of OT. If $a = 0$, then from the fact that the OT-protocol doesn't leak information about b and the fact that in this case B returns 0 to A in the final step no matter what b is, A doesn't learn nothing more about b as a result of the complete protocol.

Note that with respect to correctness and fairness, we have to assume that B indeed sends the correct value ab to A. Furthermore, we must assume here that both players take their actual choices as input to the protocol. But in any case, we can say that the protocol is secure for both parties if they are *semi-honest*. This means that both follow the rules of the game, but may try to learn as much as possible about the other player's input. We must also assume that no *crash-failures occur*, i.e. both players remain operational throughout the protocol and don't fail.

1.1 Historical Notes

Oblivious Transfer was originally introduced by M. O. Rabin [76], in a slightly different way. Namely, in his definition, the sender has just one bit b, and at the end of the protocol the receiver gets the bit b with probability 1/2. Otherwise the receiver gets "?", and doesn't receive the bit. The sender cannot tell what happened.

Even, Goldreich and Lempel [43] later defined OT as we use it here, except that they require the selection bit to be random. It turned out that Wiesner [80] had earlier devised a similar definition in unpublished work.

The definition used here has appeared in many works on OT.

Soon after the invention of OT by Rabin, M. Blum [16] has conceived *coin-flipping over the phone* and *certified electronic email* as applications of OT.

2 Variations and Other Applications of OT

The Match-Making protocol is in fact just a toy example. Oblivious Transfer is an important primitive with many powerful applications, as we shall see.

2.1 OT of Strings

Suppose that instead of bits b_0 and b_1, the sender in OT has two strings $x_0, x_1 \in \{0, 1\}^n$. Can we perform an OT where the sender receives the string x_0 if his selection bit s is 0 and the string x_1 otherwise? Note that "bitwise" OT of the n

pairs of bits x_0^i, x_1^i of x_0 and x_1 is clearly not an option, since a cheater can for instance learn bits of both strings, which contradicts the requirements of string OT (whose definition is the obvious extension of the definition of OT of bits).

A general approach to this problem of oblivious transfer of strings is due to Brassard, Crépeau and Sántha and appears in [17]. They define *zig-zag functions*. Consider a function $h : \{0,1\}^m \to \{0,1\}^n$, and for an arbitrary subset J of $\{1,\ldots,m\}$ and arbitrary $y \in \{0,1\}^m$, let y_J denote $y = (y_1,\ldots,y_m)$ restricted to the $|J|$ bits y_i with $i \in J$.

A function h is a zig-zag if for any $I \subset \{1,\ldots,m\}$, there is a $J \in \{I, I^c\}$ such that for any $x \in \{0,1\}^n$ and for a uniformly random chosen h-pre-image y of x, y_J gives no information about $h(y) = x$.

In other words, for any fixed subset of the m bits of y, it holds that either this subset of the bits or the remaining bits give no information about $h(y) = x$, and which of the two cases actually hold, does not depend on x or y.

Given such a function h, this is how one can perform chosen one-out-of-two oblivious transfer of n-bit strings x_0 and x_1. First the sender selects random y_0 and y_1 such that $h(y_0) = x_0$ and $h(y_1) = x_1$. Say that the receiver wishes to get the string x_s. Then they execute for $i = 1 \ldots n$ the protocol $\mathrm{OT}(y_0^i, y_1^i, s)$. As a result, the receiver gets all bits of y_s, applies h to it and gets x_s. Clearly, the sender has no information about s.

Let's see why it is true that at least one of the strings x_0, x_1 remains as unknown to the receiver as before the protocol. It is clear, by inspection of the protocol and the properties of OT, that even if the receiver deviates from the steps above there is some $I \subset \{1,\ldots,m\}$ such that he receives at best $y_{0,I}$ and y_{1,I^c}. Let J, with $J = I$ or $J = I^c$, be as in the definition of a zig-zag function. Then the receiver obtains at best $y_{0,J}$ and y_{1,J^c} and he has no information about $h(y_0) = x_0$, *or* obtains y_{0,J^c} and $y_{1,J}$ and he has no information about $h(y_1) = x_1$, since J only depends on h and I.

Constructions of zig-zag functions can be based on linear codes. It is easy to see that it is sufficient to construct a binary matrix with n rows such that for any subset I of the columns it holds that I or I^c has maximal rank n. Finding preimages can be done efficiently using basic linear algebra. Here is a small example with $n = 2$ and $m = 3$: the first column has entries 1 and 0, the second 1 and 1, and the third 0 and 1. Examples for larger values of n can be found for instance using recursion in combination with Vandermonde matrices, working over extension fields [17].

2.2 Oblivious Common String Verification

We describe a nice application of oblivious string transfer due to Fagin, Naor and Winkler [44]. There are two players A and B, and each of them holds some secret n-bit string. Their goal is to obliviously verify whether those strings are equal: as a result both of them should learn whether the strings are equal, but nothing more than that. Obviously, a secure protocol for this task can be used as a means of identification in a number of scenarios.

This is how the FNW protocol works. A has $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ and B has $y = (y_1, \ldots, y_n) \in \{0,1\}^n$ as private input. For $i = 1 \ldots n$, A selects random k-bit strings $r_{i,0}$ and $r_{i,1}$, and B selects random k-bit strings $s_{i,0}$ and $s_{i,1}$. The parameter k is a security parameter. In the following, if u and v are n-bit strings, then $u + v$ denotes the n-bit string whose i-th bit is equal to the sum (modulo 2) of the i-th bit of u and the i-th bit of v, $i = 1 \ldots n$.

Consider the bit strings $\alpha = \sum s_{i,x_i}$, $\beta = \sum r_{i,y_i}$, $\alpha' = \sum r_{i,x_i}$ and $\beta' = \sum s_{i,y_i}$. If $x = y$, then clearly $\alpha + \alpha' = \beta + \beta'$. Otherwise, these values are different with probability $1/2^k$ (so in order for the error to be small, k must be large).

Note that A and B can obtain the strings α, resp. β by one-out-of-two string OT. The values α' and β' can be computed by A, and B respectively from their own random choices and their input strings. This is the complete protocol.

A		B
In: $x \in \{0,1\}^n$		**In**: $y \in \{0,1\}^n$
For $i = 1 \ldots n$,		For $i = 1 \ldots n$,
$r_{i,0}, r_{i,1} \in \{0,1\}^k$: *random.*		$s_{i,0}, s_{i,1} \in \{0,1\}^k$: *random.*
	$\xleftrightarrow{A \to B \text{ : For } i = 1 \ldots n, \mathrm{OT}(r_{i,0}, r_{i,1}, y_i)}$	
	$\xleftrightarrow{B \to A \text{ : For } i = 1 \ldots n, \mathrm{OT}(s_{i,0}, s_{i,1}, x_i)}$	
$\alpha = \sum_{i=1}^n \mathrm{OT}(s_{i,0}, s_{i,1}, x_i)$		$\beta = \sum_{i=1}^n \mathrm{OT}(r_{i,0}, r_{i,1}, y_i)$
$\alpha' = \sum_{i=1}^n r_{i,x_i}$		$\beta' = \sum_{i=1}^n s_{i,y_i}$
	$\xrightarrow{\alpha + \alpha'}$	
	$\xleftarrow{\beta + \beta'}$	
Out: $\alpha + \alpha' \stackrel{?}{=} \beta + \beta'$		**Out**: $\alpha + \alpha' \stackrel{?}{=} \beta + \beta'$

Note that if one of the parties is honest, and the other party has some y_i that differs from x_i, then the latter receives a completely random string in the final exchange. There exists a variety of other solutions to this particular problem. See [40] for a more efficient solution based on OT. Both [44] and [40] additionally survey completely different approaches not based on OT.

2.3 A Reduction

Crépeau [36] has shown that the OT as defined by Rabin (Rabin-OT, see Section 1.1) and chosen one-out-of-two OT are the same in the sense that one can be simulated from the other in a blackbox fashion, vice versa.

Given chosen one-out-of-two OT as a subroutine, Rabin-OT can be simulated as follows. The sender in Rabin-OT has a bit b to be obliviously transferred. First, the sender selects bits b_0 and b_1 at random such that $b_0 \oplus b_1 = b$, and the receiver selects a random selection bit s. After they have executed $\mathrm{OT}(b_0, b_1, s)$, the sender selects a random bit t and sends (t, b_t) to the receiver. With probability $1/2$, t is different from s and hence the receiver obtains both bits b_0 and b_1 and computes $b = b_0 \oplus b_1$. Also with probability $1/2$, the receiver gets the bit he already had, leaving him in the dark about the other bit by the properties of chosen one-out-of-two OT and hence about b. The sender doesn't know what happened, since he doesn't learn s.

The interesting case is to simulate chosen one-out-of-two OT from Rabin-OT. So let the sender have input bits b_0 and b_1, and let the receiver have a selection bit s. Furthermore, Rabin-OT is at their disposal as a subroutine.

The sender chooses k random bits $\delta_1, \ldots, \delta_k$, where k is a*security parameter*. This value should be chosen large enough so that some error probability (to become clear later on) is small enough to be acceptable.

Next, using Rabin-OT, the sender transmits these bits one by one to the receiver, who is expected to receive roughly half of them. It is important to note that with probability $1 - 1/2^k$, at least one of the bits is not received, and with the same probability some bit is received.

Let $I \subset \{1, \ldots, k\}$ denote the collection of j such that δ_j has been received. Likewise, I^c refers to the bits that have not arrived. Having selection bit s, the receiver writes $I_s = I$ and $I_{1-s} = I^c$, and sends the ordered pair (I_0, I_1) to the sender. The sender now knows that the receiver obtained the bits corresponding to I_0 *or* I_1, but both events are equally likely from his point of view by the properties of Rabin-OT. Next, the sender adds all bits $\delta_i, i \in I_0$ to b_0 and the bits $\delta_i, i \in I_1$ to b_1, and sends the resulting two bits to the receiver. Since the latter knows all bits $\delta_i, i \in I_s$, he can recover the bit b_s, as required. It's clear that the sender has no clue about s.

Finally, consider a cheating receiver, who might define the sets I_0, I_1 differently, and perhaps learn more. However, if I_0 and I_1 cover the full set $\{1, \ldots, k\}$, then with probability $1 - 1/2^k$ at least one of the sets, say I_0, contains an index referring to a bit not received, which is hence completely unknown. In this case, the receiver doesn't learn b_0.

Sender		Receiver
In: $b_0, b_1 \in \{0,1\}$ $\delta_1, \ldots, \delta_k \in \{0,1\}$: *random*		**In**: $s \in \{0,1\}$
	$S \to R$: Rabin-OT$(\delta_1), \ldots,$ Rabin-OT(δ_k) $\longleftrightarrow$	
		received: $\delta_j, j \in I$ set $I_s = I$, $I_{1-s} = I^c$
	$\xleftarrow{I_0, I_1}$	
$I_0 \cup I_1 \stackrel{?}{=} \{1, \ldots, k\}$ $z_0 = (\oplus_{i \in I_0} \delta_i) \oplus b_0$ $z_1 = (\oplus_{i \in I_1} \delta_i) \oplus b_1$		
	$\xrightarrow{z_0, z_1}$	
		Out: $b_s = (\oplus_{i \in I_s} \delta_i) \oplus z_s$

3 Constructions of OT-Protocols

For OT protocols to exist, we *must* make assumptions about the world in which the players operate, for instance related to the communication channel connecting the players, or their computational abilities.

However, besides its elegance and usefulness in protocol design, it is interesting to note that OT can be implemented under a wide variety of different such assumptions.

Among these, the difficulty of factoring large random composite integers, the Diffie-Hellman problem (related to the difficulty of computing discrete logarithms), and abstract, general assumptions such as the existence of trapdoor one-way permutations (which can be implemented under the RSA assumption) [59]. But physical assumptions suffice as well, such as the OT based on noisy communication channels of Crépeau and Kilian [39].

3.1 Necessity of Assumptions

Why doesn't OT exist unconditionally? Indeed suppose that a protocol for OT exists, making no assumptions on the computational abilities of the players, the communication channel or whatever.

Then there are programs used by sender and receiver to compute the exchanged messages, that given random strings and the input bits would operate deterministically. Moreover, we may assume that the players communicate over a perfect channel, and that the players have infinite computing power. Say that the protocol achieves perfect correctness and always halts.

This leads to a contradiction as shown by the following (informal) argument, even if we assume that the players are semi-honest.

Given OT, there exists a protocol with similar characteristics for two players A and B to obliviously evaluate the AND of their input bits a and b (see Section 1). We show that such a protocol does not exist.

Let $\mathcal{T}$ denote the sequence of messages exchanged in a completed execution of the protocol ($\mathcal{T}$ is called *transcript*). Write $x_A \in \{0,1\}^*$ and $x_B \in \{0,1\}^*$ for the respective random strings used by A and B in the computation.

Say that $a = 0$. We argue that a semi-honest A having $a = 0$ as input can always figure out the value of B's input b, thus contradicting the security conditions.

If $b = 0$, then there exists a random string $x'_A \in \{0,1\}^*$ such that $\mathcal{T}$ is consistent with A having input $a = 1$ instead of $a = 0$. This follows from the fact that B, having $b = 0$ as input, has no information about A's input. Clearly, fixing the transcript $\mathcal{T}$ and an arbitrary x'_A, and setting $a = 1$, A can effectively decide whether the transcript is consistent with x'_A and $a = 1$. Since we do not assume limits on the computational power of the players, A eventually finds such string x'_A.

In case that $b = 1$, it is clearly impossible that $\mathcal{T}$ is consistent with $a = 1$ and some x''_A, since in this case flipping A's input from 0 to 1, changes the logical AND of the inputs: since we assumed perfect correctness, $\mathcal{T}$ cannot be consistent with two pairs of inputs (a,b) whose respective logical AND is different.

Therefore, A decides that $b = 0$ if there exists x'_A such that $\mathcal{T}$ is consistent with x'_A and $a = 1$, and decides that $b = 1$ if no such x'_A exists.

Similar arguments apply to the OR-function. Based on information-theory, one can find a more general argumentation.

3.2 Rabin-OT

We present a version of the original Rabin-OT [76]. Let n be the product of two distinct, large random primes p and q. By the assumption that factoring large random composite integers is infeasible [2], it is hard to retrieve p and q given just n.

However, it's easy to generate such n with known factorization. Testing primality can be done efficiently [3], and by the Prime Number Theorem, the fraction of primes smaller than K is roughly $1/\ln K$ for large K. Therefore, one can just select a random large integer and test it for primality. After some tries one finds a random integer that one knows is prime. Multiplying two such primes gives n.

[2] though certainly not *impossible*.

[3] i.e., certainly in practice. There is also a theoretical result by Adleman and Huang, extending a result by Goldwasser and Kilian, saying that primality can be tested in probabilistic polynomial time, with a negligible probability that no decision is made.

Rabin-OT is based on the number-theoretic fact that given two square roots x and z of a square y modulo n, that do not differ by a sign, one can efficiently compute p and q from those roots and n. Indeed, from $x^2 \equiv z^2 \bmod n$ we get $(x+z)(x-z) \equiv 0 \bmod n$. And since $x \not\equiv \pm z \bmod n$, n doesn't divide $(x+z)$ and doesn't divide $(x-z)$, yet it divides $(x+z)(x-z)$. This is only possible if p divides exactly one of the two terms, and q divides the other. We now just compute the greatest common divisor of n and $(x-z)$ and the greatest common divisor of n and $(x+z)$, to get both factors p and q. Note that greatest common divisor can be efficiently computed using for instance Euclid's algorithm.

Each square y modulo n has four distinct square roots. Indeed, modulo each of the factors p and q, there are two square roots. Combining them with the Chinese Remainder Theorem, we get 4 distinct roots modulo n.

From the difficulty of factoring, and the analysis above, we conclude that that squaring modulo n is a *one-way function*, i.e. given just n and a random square y modulo n, it is infeasible to find a square root of y. Indeed, if this were not so, then one would select a random x, compute y as the square modulo n of x and compute a square root z of y given just y and n. With probability $1/2$, $x/z \bmod n$ is a non-trivial root of 1, and one can factor n efficiently.

On the other hand, if one knows p and q, computing a root of a square is efficient. It's easy to explain in the case that p and q are both $3 \bmod 4$. Let y be a square modulo p, and write $z^2 \equiv y \bmod p$. Define $x \equiv y^{(p+1)/4} \bmod p$. Then $x^2 \equiv y^{(p+1)/2} \equiv z^{p+1} \equiv z^2 \equiv y \bmod n$. Same story for computing square roots modulo q. So if one has a square modulo n and one knows p and q (both of them $3 \bmod 4$), one projects the problem modulo n on the factors p and q, computes square roots, and lifts it back with the Chinese Remainder Theorem. If p and q, are not both $3 \bmod 4$, it's more complicated. We say that squaring modulo n is a *trapdoor* one-way function.

Without giving further details, we state that it is possible to encode a bit b as an integer modulo n using a public function $\mathrm{ENCODE}(b, n)$, such that it is hard to retrieve b given just $\mathrm{ENCODE}(b, n)$ and n, but easy given the trapdoor for n as well, i.e. its factorization.

The protocol works as follows. The sender encodes the bit b that is to be sent by computing $\mathrm{ENCODE}(b, n)$. After receiving n and $\mathrm{ENCODE}(b, n)$ from the sender, the receiver selects a random x modulo n and sends its square y modulo n to the sender. Note y perfectly hides which out of the four possible roots the receiver has chosen. The sender, knowing p and q, can efficiently compute a random square root z of y and returns it to the receiver. With probability $1/2$, z does not differ by a sign from x, and the receiver can factor n, and efficiently retrieve b from $\mathrm{ENCODE}(b, n)$. Otherwise, also with probability $1/2$, $z \equiv \pm x \bmod n$, and the receiver doesn't get the factorization of n, and hence doesn't get closer to learning b.

Sender		Receiver
In: $b \in \{0,1\}$		
	$\xrightarrow{n,\,\text{ENCODE}(b,n)}$	
		$x \in \mathbb{Z}_n^*$: *random*
		$y \equiv x^2 \bmod n$
	$\xleftarrow{\;\;y\;\;}$	
z : random *s.t.*		
$z^2 \equiv y \bmod n$		
	$\xrightarrow{\;\;z\;\;}$	
		If $z \not\equiv \pm x$,
		factor n, get b
		Out: b
		Else **Out**: ?

It is assumed that both players are semi-honest. For sender security we have to assume that the receiver is computationally bounded. The security of the receiver is unconditional.

3.3 OT based on RSA

We give an example for chosen one-out-of-two OT based on RSA [77], the well-known public-key encryption scheme which R. Rivest, A. Shamir and L. Adleman introduced in 1978. We assume that both players are semi-honest. The sender selects two large random distinct primes p and q, and computes $n = pq$, the *modulus*. Next, the sender selects an integer *exponent* e such that e is relatively prime to $(p-1)(q-1)$. Let the integer d satisfy $de \equiv 1 \bmod (p-1)(q-1)$ (given p, q and e such d is easy to compute). Now we have $(x^e)^d = (x^d)^e \equiv x \bmod n$ for all x. The sender keeps d secret, and sends n, e (public key) to the receiver.

It has been proved by Alexi, Chor, Goldreich and Schnorr [1] that if a plaintext x is chosen at random, guessing the least significant bit of x, given just the ciphertext $y = x^e \bmod n$, n and e, significantly better than at random, is as hard as finding all bits of x. This is called a *hard-core bit* for the RSA function. Note that this result does not follow directly from the usual RSA-security assumption. That assumption only states that it is infeasible to recover *all* bits of x from y. In the protocol to follow, the sender in OT exploits the existence of hard-core bits to "mask" his bits b_0 and b_1.

The receiver, having selection bit s, chooses a random plain text $m \bmod n$ and computes the cipher text $c_s \equiv m^e \bmod n$. Let r_s denote the least-significant bit of the plain-text m.

The receiver selects the ciphertext c_{1-s} as a random integer modulo n and communicates the pair of ciphertexts (c_0, c_1) to the sender. The sender, knowing the secret RSA-key, computes for each of those ciphertexts their respective least-significant bits r_0 and r_1. Now the sender masks the bits b_0 and b_1 by setting

$b'_0 = b_0 \oplus r_0$ and $b'_1 = b_1 \oplus r_1$, and sending them to the receiver. The receiver recovers b_s by computing $b'_s \oplus r_s$. The bit b_{1-s} remains concealed, since he cannot guess r_{1-s} with high enough probability. Note that the selection bit s is unconditionally hidden from the sender, and that we have to assume that the receiver is semi-honest in order to guarantee sender security.

This is essentially the OT protocol of Goldreich, Micali and Wigderson [59], which not only works for RSA but any other trapdoor one-way permutation as well (though in general, more care has to be taken to define a hard-core bit).

4 General Secure Two-Party Computation

It is a natural question to ask which functions other than AND or string equality can be obliviously evaluated. It is the answer to this question that demonstrates the power of oblivious transfer: *all* functions f with finite domain and finite image can be obliviously evaluated. This is due to A. Yao [81], who based his result on the assumption that factoring integers is intractable. The protocol below shows the stronger result saying that the existence of OT is sufficient for this task. This is due to O. Goldreich and R. Vainish [60].

For simplicity, think of a function $f : \{0,1\}^{n_A} \times \{0,1\}^{n_B} \to \{0,1\}$, where n_A and n_B denote the number of input bits player A and B hold.

The function f is assumed to be efficiently computable (polynomial time on a Turing-machine) and both players have agreed on a *Boolean circuit* computing f (so in particular they both know f):

- a directed acyclic graph with
- $n_A + n_B$ input nodes, and one output node.
- The remaining vertices are labelled as binary negation, and two-input binary addition and multiplication gates. Note that these operations correspond to binary NOT, XOR and AND. The outputs of internal gates can be led to an arbitrary number of other gates (arbitrary fan-out).
- The topology of the graph dictates the flow of the values on which the computations are performed. More precisely, the circuit computes f in the sense that if one assigns the bits of any input strings a, b to the input nodes, and inductively propagates the values resulting from the computations performed on them (according to the logic of the gates), then the output node will be set to $f(a, b)$.

It is well known that all computable functions f can be computed by Boolean circuits and that a Boolean circuit computing f can be constructed with a number of nodes (gates) polynomial in the number of inputs (i.e. $n_A + n_B$ in this case) if f is efficiently computable.

The problem of oblivious function evaluation of f is as follows. Player A has input $a \in \{0,1\}^{n_A}$, and player B has input $b \in \{0,1\}^{n_B}$. For fixed input a, b, and a fixed circuit computing f, the computation graph is the graph representing the circuit but with the flow of the values written on the edges. For a given gate in the computation graph, we speak of the *actual inputs* and the *actual output*.

We try to devise a protocol for A and B to execute such that both learn $f(a, b)$, but none of them learns more than what is implied by $f(a, b)$ and the own input. In the following we assume that neither player crashes, and that both of them are semi-honest.

The protocol consists of three stages.

Input Sharing. For each of the n_A bits a_i of his input string a, A selects two bits $s_{i,A}$ and $s_{i,B}$ at random such that $s_{i,A} \oplus s_{i,B} = a_i$ and sends $s_{i,B}$ to B. Player B does the same to the n_B inputs bits b_i of his input string b, resulting in $t_{i,A}$ and $t_{i,B}$. This is an *additive secret sharing of the inputs*, and the s and t values above are called *shares*.

Computation. The computation proceeds inductively and in a gate by gate manner, possibly handling many gates in parallel. The players maintain the following *invariant*. The actual inputs to the current gate are additively shared. After processing of the current gate, there are uniformly random shares u_A (held by A) and u_B (held by B) such that $u_A \oplus u_B$ equals the actual output of the current gate and such that neither player has increased knowledge about the actual output.

Output Reconstruction: Each player reveals his share in the output bit of the computation. The sum of these shares equals the output bit $f(a, b)$.

It remains to be shown how this invariant is maintained for each of the three types of gates.

4.1 Addition-Gates

Let x_0, x_1 denote the actual input bits, and let $x = x_0 \oplus x_1$ denote the actual output bit. Then A holds $x_{0,A}$ and $x_{1,A}$, and B holds $x_{0,B}$ and $x_{1,B}$ such that $x_{0,A} \oplus x_{0,B} = x_0$ and $x_{1,A} \oplus x_{1,B} = x_1$.

Player A computes $x_A = x_{0,A} \oplus x_{1,A}$ as his share in the actual output bit x of the current gate. For B there is a similar program, resulting in a share x_B. We have $x = x_A \oplus x_B$.

4.2 Negation-Gates

These are simply handled by designating one player, say A, who just flips his share in the actual input bit, and takes the result as his share in the actual output bit. B just takes his share in the actual input bit as his share in the actual output bit.

4.3 Multiplication-Gates

A more interesting case is multiplication. Again, let x_0, x_1 denote the actual input bits. Then A holds $x_{0,A}$ and $x_{1,A}$, and B holds $x_{0,B}$ and $x_{1,B}$ such that $x_{0,A} \oplus x_{0,B} = x_0$ and $x_{1,A} \oplus x_{1,B} = x_1$.

Before we proceed, let's take a look at OT once more. Suppose that player A has some bit α and that player B has some bit β. How can they arrive at the situation where they hold random additive shares in $\alpha \cdot \beta$ but neither of them has gained information about $\alpha \cdot \beta$?

Let ρ be a secret random bit chosen by player A. If A and B now execute the OT-protocol with A acting as the sender and B as the receiver, using $b_0 = \rho$ and $b_1 = \alpha \oplus \rho$, and $s = \beta$ as their respective inputs, we see that B gets the value $\alpha\beta \oplus \rho$ as output. In other words, $\mathrm{OT}(\rho, \alpha \oplus \rho, \beta) = (\beta \oplus 1)\rho \oplus \beta(\alpha \oplus \rho) = \alpha\beta \oplus \rho$. A then just takes ρ as his share, and B takes $\alpha\beta \oplus \rho$ as his share. We only need to argue that A and B do not gain knowledge about each other's inputs as a result. Clearly, the security of OT implies that A doesn't gain knowledge about β, since it is B's selection bit. Again by the security of OT, B learns only one of ρ and $\rho \oplus \alpha$, and since ρ was chosen at random by A, this doesn't increase B's knowledge about α.

Sender A	Receiver B
$\rho \in \{0,1\}$ random	
In: $\alpha \in \{0,1\}$	**In**: $\beta \in \{0,1\}$

$$\xleftrightarrow{\quad A \to B : \mathrm{OT}(\rho, \alpha \oplus \rho, \beta) \quad}$$

Out: ρ	**Out**: $\alpha\beta \oplus \rho$

Now we return to handling the multiplication gates. Note that

$$x = x_0 \cdot x_1 = (x_{0,A} \oplus x_{0,B})(x_{1,A} \oplus x_{1,B}) =$$

$$x_{0,A}x_{1,A} \oplus x_{0,A}x_{1,B} \oplus x_{1,A}x_{0,B} \oplus x_{0,B}x_{1,B}.$$

Two executions of OT with, say, A as the sender are sufficient to get to the random additive shares of x. A selects random bits ρ_{01} and ρ_{10}.

1. $A \to B$: $\mathrm{OT}(\rho_{01}, \rho_{01} \oplus x_{0,A}, x_{1,B}) = \rho_{01} \oplus x_{0,A}x_{1,B}$.
2. $A \to B$: $\mathrm{OT}(\rho_{10}, \rho_{10} \oplus x_{1,A}, x_{0,B}) = \rho_{10} \oplus x_{1,A}x_{0,B}$.

A takes as his share in x the bit $x_A = x_{0,A}x_{1,A} \oplus \rho_{01} \oplus \rho_{10}$, and B takes $x_B = x_{0,B}x_{1,B} \oplus x_{0,A}x_{1,B} \oplus x_{1,A}x_{0,B} \oplus \rho_{01} \oplus \rho_{10}$ as his share.

4.4 Complexity of the Protocol

By inspection, an upperbound on the communication costs of executing the protocol is $O(|C|)$ OT's and $O(|C|)$ bits (the latter is from the initial input sharing), where $|C|$ denotes the number of gates in the circuit computing the function f. Handling many gates in parallel, the round complexity is upper bounded by the *depth* of the circuit C, i.e. the length of the longest path in the graph of C.

4.5 Security Discussion

In order that the above oblivious circuit evaluation protocol satisfies the required *correctness* and *privacy* properties we have to assume that both players are *semi-honest*, i.e. they follow the protocol and behave exactly as required, but each of them separately may try to deduce from the information available to them as a result of the protocol execution as much as possible about the other player's inputs.

It is easy to see that if one of the players is *malicious* and deviates from the protocol, he can make the other player accept a false result, while he in fact knows the correct one. With an adequate definition of what it means for OT to be secure against malicious attacks, the protocol above would be private though. For *fairness*, we have to assume that neither player crashes before termination of the protocol.

The intuition behind the analysis of privacy is that the invariant maintained guarantees that at each point in the execution of the protocol, the players hold random additive shares in the actual outputs so far and that the respective shares of each player does not increase knowledge about the actual output so far. It is only at the end of the protocol where they have random additive shares in the actual output that are exchanged, enabling the reconstruction of the actual output.

Therefore, another way to look at the protocol is by saying that, conceptually speaking, it simulates a *trusted host*: a third party who is and can be trusted by both players. Given such a third party, both players secretly send their inputs to the host, who returns the function value to both players. This is called an *ideal protocol*.

In an actual proof, one has to show that each player on his own, given just his input and the result of the computation, is able to generate efficiently a simulation of the protocol that is indistinguishable from the ideal protocol.

Later we present protocols for the same task, that are secure against much stronger adversaries than semi-honest ones in a much broader context, and in fact, the security principles outlined above are the basis for defining security there as well (Beaver [4], Micali/Rogaway [70], Goldreich [62], Canetti [21]).

5 Example

As an illustration, let's return to the problem of Oblivious Common String Verification. We show that the general protocol provides a solution for this problem. There are good reasons to prefer the solution of Fagin, Naor and Winkler, mainly because an appropriate OT withstanding attacks by malicious rather than semi-honest players renders the complete FNW solution secure against this kind of attack.

But if we may assume the players are semi-honest, the following protocol is just as good. Two players A and B each hold some secret n-bit string. Write $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ for their respective strings.

Write $f(x_1, \ldots, x_n, y_1, \ldots, y_n) = (x_1 \oplus y_1 \oplus 1) \cdots (x_n \oplus y_n \oplus 1)$. It follows that $f(x, y) = 1$ if and only if $x = y$.

From this formula for f we can easily derive a Boolean circuit: there are n pairs of input bits (x_i, y_i). For each such pair, the bits in it are first led through a binary addition gate, after which the result is passed through a negation gate. Now there are n intermediate results, which only have to be led through an n-input binary multiplication gate. To be consistent with our description, we first write the n-input binary multiplication gate as a tree of depth $\log n$ with two-input binary multiplication gates only, and lead the intermediate results into it.

By the method from Section 4 A and B can now obliviously verify whether or not they have the same string. Note that $2n$ oblivious transfers and $2 \log n$ rounds of communication suffice.

6 Dealing with Malicious Attacks

Unfortunately, most protocols presented so far only work if the players are semi-honest. We first indicate the failures that occur in the examples we have given, if one of the players is cheating and deviates from the protocol, i.e. carries out a *malicious* attack. The rest of this section deals with methods to enhance the security of OT-protocols, achieving security even in the presence of a malicious attacker. We stress that we still assume that the players do not crash before the end of the protocol, to ensure fairness.

As an example of a failure, although Rabin-OT is secure for the sender if the receiver is semi-honest and factoring large integers is hard, it is not clear that a receiver *deviating* from the steps required in the protocol couldn't "extract" the factorization from the sender, even without being able to factor large numbers efficiently *in general*. It might be true that there exists a single number modulo n such that a square root of it reveals the factorization of n. Given such a number it would be easy for the receiver to get the factorization of the sender's modulus, since the sender returns a square-root of *any* number modulo n that the receiver sends. Hence, the receiver would always get the bit b. On the other hand, if the sender would choose the modulus n as the product of three primes for instance, he can influence the probability with which the receiver gets the bit b.

Fischer, Micali and Rackoff [48] presented the first realization of OT *secure against malicious attacks*, i.e. it provides security for sender and receiver even if one of them deviates arbitrarily from the protocol.

It is easy to see that the scheme based on RSA we presented is totally insecure against malicious attacks by the receiver: nothing prevents the receiver from computing the ciphertext c_{1-s} in the same fashion as c_s, in which case the receiver retrieves both b_0 and b_1 at the end of the OT.

6.1 Notion of Security of Basic OT

We assume that at most one of the players carries out a malicious attack. These are the minimum (and for some applications sufficient) requirements we have to make in order for OT resisting malicious attacks to make sense.

1. If the sender is honest (so the bits b_0 and b_1 are well-defined) throughout the protocol, "no matter" how the receiver plays (note that if the receiver is corrupt, the selection bit s may not even be well-defined in general), at least one of the bits b_0, b_1 remains "completely" unknown to him. [4]
2. If the receiver is honest throughout the protocol, "no matter" how the sender plays, the selection bit s remains "completely" unknown to the sender. Moreover, the receiver always gets some bit, or else just aborts and the sender is deemed corrupt.

Under this definition, the string-equality protocol of [44] as presented in Section 2.2 is secure against a malicious attack by one of the players, for instance.

Beaver [9, 10] has a simulation based definition of secure OT.

6.2 A General Solution in the Cryptographic Scenario

Goldreich, Micali and Wigderson [59] have a general defense against malicious attacks that works in principle for any OT based on intractability assumptions. We give an informal overview. It involves three other important primitives: *commitment schemes*, *mutually random coins* and *general zero knowledge techniques*. Interestingly, all these primitives (including OT) can be realized under the assumption that trapdoor one-way permutations exist.

Trapdoor One-Way Permutations. We assume that both players are restricted to probabilistic polynomial time computations, so that none of the players is computationally powerful enough to invert one-way permutations without knowing a trapdoor. More precisely, this means that if a trapdoor one-way permutation is selected at random by one party, then the other party, having access to the description of the forward function only, cannot efficiently invert a randomly chosen element from its range. The party knowing the trapdoor can efficiently invert the function. Why is it that one party does have the trapdoor while the other doesn't? This is by the existence of a special probabilistic polynomial time algorithm called *trapdoor permutation generator*. On input of a random bit string, the generator outputs a "random" one-way permutation *and* a corresponding trapdoor. RSA (see Section 3.3) is an example of a trapdoor one-way permutation.

[4] Actually, one must require that there is a bit s so that if b_s is *given* to the receiver, he still has no information about b_{1-s} This is to exclude the possibility that the receiver for instance learns $b_0 \oplus b_1$

Commitments. Conceptually, there is an analog between vaults and commitment schemes. Player A has some secret piece of information, and places the piece in a vault, locks it and memorizes the opening combination. He then passes the vault on to player B, to whom the secret information is *hidden* until he gets the secret opening information to open the vault. But in the mean time, player A cannot change the information stored in the vault, since it is no longer in his possession. Thus, the commitment is *binding*. At some later moment, player A can simply send the key of the vault to player B, who can then *open* it and read the information.

Cryptographic, non-physical realizations of commitment schemes, can for instance be based on RSA. Player A generates a key-pair $((n, e), (p, q))$ for RSA, and sends the public-key to player B. To commit to a bit b, A generates a random plaintext m, and computes the corresponding ciphertext c. Write ρ for its least significant bit. He sets $d = b \oplus \rho$ and sends (c, d) as the commitment to B. To open the commitment, A sends m and the bit b to B, who verifies that m is the plaintext corresponding to c and that d is equal to the sum of its least significant bit and b. The hiding property follows from the fact that the least significant bit is a hard-core bit (see Section 3.3). The commitment is binding since RSA is a permutation (if the public exponent e is a prime larger than the modulus n, for instance, B can efficiently verify that the public key defines a permutation without any further proofs from A, since then we have $\gcd((p-1)(q-1), e) = 1$ for sure and primality can be efficiently tested). Note that the binding property is unconditional and that the hiding property holds if B is polynomially bounded. In fact, it can be shown that *one-way permutations* are sufficient for commitments.

This seemingly innocent primitive has far reaching applications in cryptography. For instance, it is sufficient to implement general zero knowledge interactive proofs [58, 61], a method that allows one to prove "anything provable" in zero knowledge, i.e. to convince a sceptical judge of the veracity of an assertion without giving anymore information away than the fact that the assertion is true. [5]

Mutually Random Coins. Another application of commitments is mutually random coins. Here players A and B want to establish a bit (or a string) that is random if one of them is honest. A simple protocol goes as follows. A selects a random bit b_A and sends to player B a commitment to it. Player B selects a random bit b_B and sends it to A, who opens the commitment. The bit b is defined as $b = b_A \oplus b_B$.

OT Secure against Malicious Attacks. Returning to the problem of defending against malicious attacks in OT, we now show how we can defend against these attacks by the techniques of [59].

[5] There is a vast literature dealing with general zero knowledge and commitment techniques, with many different flavours, styles and security and efficiency properties, but we do not discuss these any further here.

The key observation is that, for instance in the RSA based example of OT, if one of the players is honest the security of the other player is guaranteed. That's not what we want, since actually we want that a player's security is guaranteed if he is honest, no matter what the other player does. Nevertheless, in some sense this fact is the basis for achieving it: based on the primitives outlined above, an honest player can force the other player to be honest as well or else the protocol simply halts with no advantage for the corrupt player.

This has become an important design principle throughout the field of cryptography: often it is possible to start from a cryptographic protocol that is secure if its participants are semi-honest and to transform it into a protocol secure against malicious adversaries, by forcing each player to *prove* that he behaved as a semi-honest participant.

We start looking into the details. First of all, it's useful if the randomness used by each player is mutually random. However, it is in the interest of both players not to reveal their randomly chosen bits, for obvious security reasons. Say that each player needs at most l random bits. Then they execute the protocol for achieving a mutually random bit l times in parallel where the receiver is the committing party, and l times in parallel where the sender is the committing party. However, they do not open any of the commitments used. Note that in the first case this implies that the receiver knows the resulting mutually random bits whereas the sender does not. So the receiver can use these bits later on whenever they are required, and in fact we will explain how the sender can verify that the receiver used them, in a way that is secure for the receiver. The second case is of course similar, with the roles reversed.

Let's first look at the sender's security and let's look at the RSA-example from Section 3.3. The sender wants to make sure that the receiver gets at most one of the bits b_0, b_1. It is sufficient if the receiver can convince the sender of the veracity of the following assertion about the ciphertexts c_0, c_1. One of them is equal to some particular string of mutually random bits, and one of them is equal to the RSA-function applied to some other particular string of mutually random bits (in this case we refer to those mutually random bits that the receiver knows, but the sender doesn't). To protect the receiver in case he is honest, the means by which the receiver convinces the verifier of this assertion must be zero-knowledge.

Roughly speaking, it is now fairly easy though tedious for the sender and receiver to efficiently derive by themselves from what is known to both of them, a description of a function F and a function-value y such that the assertion about the ciphertexts c_0, c_1 is equivalent to saying that there exists an x with $F(x) = y$. Furthermore, if the receiver followed the protocol, he can actually efficiently determine such x.

The zero knowledge techniques from [58] are designed for exactly this technical situation! So in principle, the security of the sender can be guaranteed.

As to the receiver's security, his selection bit is protected by the fact that the proof of the assertion is zero knowledge.

Therefore, under fairly general intractability assumptions, OT that is secure against malicious attacks can be realized. However, it is very costly to resort to the powerful techniques underlying the defense. In concrete situations with specific implementations of OT, there may exist a more efficient way to enhance the security.

Oblivious Function Evaluation and Malicious Attacks. So, are we done and can we now use the OT with enhanced security directly in the general protocol from Section 4 and obtain general oblivious function evaluation secure against malicious attacks by one of the players?

No! There are many more things to be fixed first. For instance, in each current gate, the inputs used must be the same as the outputs of some earlier gates. Here a solution is to have both players always commit to their inputs at each current gate and have them prove to each other in zero knowledge that these commitments commit to the same values as the outputs of the gates that are supposed to deliver the inputs to the current gate. In particular, both players commit to their initial inputs.

Furthermore, using similar techniques as in the case of OT with strengthened security above, it is not so difficult anymore to handle the full set of instructions from Section 4 securely at all gates.

We return later to the techniques of GMW [59].

7 A Generic Solution

Another fundamental result is by Kilian [67], who shows constructively that OT is necessary and sufficient for general oblivious function evaluation, *even if* one of the players is malicious. From the previous section it should have become clear that this is by no means obvious.

Given OT as a black-box [6] and given a function to be obliviously evaluated and a circuit for it, there is a generic transformation that results in a set of protocols for the players to execute. It is immaterial how the OT works exactly: at those points in the protocols where OT is required, only calls are made to a black-box for doing OT. In the other direction, note that OT can be viewed as a oblivious evaluation of the function $f(b_0, b_1, s) = (s \oplus 1)b_0 \oplus sb_1$.

Another contribution of [67] concerns the round-complexity of general secure function evaluation, which is shown to be constant with polynomial size message complexity if the function can be computed by a polynomial size *formula* (i.e. the fan-out of the gates in the circuit is 1).

We do not overview a proof of Kilian's result, but only introduce some of its fundamental parts.

[6] It is beyond the scope of the present paper to discuss the exact security definition required for this result.

7.1 Commitment based on OT

Kilian shows how a commitment scheme (which he attributes to Crépeau) can be simulated from OT as follows. Let b be the bit that player A wants to commit to. A and B agree on a security parameter k.

1. For $i = 1 \ldots k$, A selects a pair of random bits (r_i^0, r_i^1) such that $b = r_i^0 \oplus r_i^1$.
2. For $i = 1 \ldots k$, B selects a random bit s_i.
3. For $i = 1 \ldots k$, with A being the sender and B the receiver, they execute $OT(r_i^0, r_i^1, s_i)$
4. B takes the bits received, together with his own random choices, as A's commitment.
5. To open the commitment, A reveals the bit b and, for $i = 1 \ldots n$, the ordered pairs (r_i^0, r_i^1). Player B accepts the opening if and only if this information is consistent.

Player A's cheating probability is at most $1/2^k$: if A were able to open the commitment in two different ways, he would have to guess all of B's random bits, so the binding property is satisfied. The hiding property follows immediately from the definition of OT.

7.2 Committed Oblivious Transfer (COT)

COT is as OT, except that

1. Initially, the sender is committed to his input bits b_0, b_1, and the receiver is committed to his selection bit s.
2. At the end of the protocol, the receiver is committed to the received bit b_s.

An alternative proof of Kilian's result can be found in [38], who introduce COT and show that it is sufficient for secure function evaluation tolerating a malicious attacker, and that COT can be simulated from OT (they don't treat the constant round issue though). The latter construction involves so-called envelope techniques and error correcting codes.

8 Other Work

Some suggestions for further reading about defining OT secure against malicious attacks and constructions of secure OT: Beaver addresses the pitfalls in attempts to define OT secure against malicious attacks and presents solutions and constructions [7, 9]. In [10], he gives a precise definition of OT so that when used in a multi-party computation protocol the protocol as a whole is secure against a malicious *adaptive* attacker.

The above references and Crépeau [37] (besides those references we already mentioned) contain a host of other interesting references.

Part II

General Secure Multi-Party Computation

9 Introduction

The protocols from Sections 4 and 6.2 have an obvious extension from two players to n players guaranteeing correctness and privacy. This is done by using n-out-of-n additive sharing of bits and executions of OT between every pair of players. In this case, *privacy* of a single player is guaranteed even if the $n-1$ other players pool their complete views on the protocol. The extension to $n > 2$ players of the protocol from Section 6.2 is even secure against malicious attacks.

However, the fairness condition is only fulfilled by making a strong assumption on the behaviour of the players, since one party can leave the protocol knowing the result of the computation whereas the other remain ignorant about it, or simply disrupt it in an early stage.

An important contribution of Goldreich, Micali and Wigderson [59], is that they explain how *privacy* can be traded for *fairness*. In fact, they achieve the stronger property of *robustness*: it is not only infeasible for corrupted parties to walk away prematurely with the result of the computation and leaving the remaining players ignorant about it, they can't disrupt the computation at all: if the corrupt players leave the computation, the remaining ones will still be able to complete the computation.

More precisely, they show that even if at most a *minority* of the players perform a coordinated malicious attack, then correctness, privacy *and* robustness can be guaranteed.

Apart from GMW-techniques we discussed in Section 6.2, they employ what is called *verifiable secret sharing*, which was first introduced by B. Chor, S. Goldwasser, S. Micali and B. Awerbuch [27].

Before sketching the full protocol of GMW, we introduce secret sharing and verifiable secret sharing in the next sections.

10 Secret Sharing with Semi-Honest Participants

In a secret sharing scheme there is a *dealer* and a number of *agents*. The dealer holds some secret string s, and sends *shares* of s privately to each of the agents. These shares are computed in such a way that only certain specific subsets of the agents can *reconstruct* the secret s by pooling their shares, while others have no information about it.

Secret sharing was invented independently by A. Shamir [79] and B. Blakley [15] in 1979. Their solutions allow the dealer to consider any number n of agents and any *threshold* $t \leq n$, such that from any subset of size at least t of the shares the secret s can be reconstructed uniquely and efficiently, whereas sets containing less than t shares contain no information at all about s.

We explain Shamir's scheme which is based on *Lagrange-interpolation* over finite fields. We assume that all parties involved are *semi-honest*.

10.1 Lagrange Interpolation

We use the following version of Lagrange Interpolation. Let K be a (finite) field.

Let V be a finite set of indices, and write $|V| = l$. Suppose we are given a collection of l points (p_i, q_i), $i \in V$, in the plane K^2, where the p_i's are all different. Then there is a unique polynomial $f(X) \in K[X]$ of degree smaller than l, that passes through these l points, i.e. $f(p_i) = q_i$ for all $i \in V$.

First we discuss existence. For each $i \in V$ define the polynomial

$$f_{V,i}(X) = \prod_{j \in V \setminus \{i\}} \frac{(X - p_j)}{(p_i - p_j)}.$$

Observe that each $f_{V,i}$ has degree exactly $l - 1$ and that $f_{V,i}(p_i) = 1$ whereas $f_{V,i}(p_j) = 0$ if $j \neq i$.

But then it follows immediately that the following polynomial f does the trick (Lagrange interpolation formula).

$$f(X) = \sum_{i \in V} q_i \cdot f_{V,i}(X).$$

Note that $f(X)$ has degree *at most* $l - 1$. Indeed, it can be strictly smaller than $l - 1$.

As to uniqueness, note that if there were a polynomial $f'(X) \in K[X]$ of degree smaller than l that agrees with f on all l points, then the polynomial $f - f' \in K[X]$ has l zeroes while its degree is smaller than l. So $f - f'$ must be identical to the zero-polynomial, since it's well known that any polynomial $g \in K[X]$ has at most degree(g) zeroes unless it's the zero-polynomial.

10.2 Shamir's Scheme

To set up Shamir's secret sharing scheme, let K be a finite field with $|K| > n$, where n is the number of agents. Let $P_1, \ldots, P_n$ be distinct, non-zero elements of K, and let these values serve as "names" for the n agents. Let $1 \leq t \leq n$ be the threshold. The secret-space in which the dealer codes the secret s is K. For each $s \in K$, define $\Pi(t, s)$ as the set of all polynomials $f(X) \in K[X]$ such that degree$(f) < t$ and $f(0) = s$.

One can efficiently sample a random member from $\Pi(t, s)$ by setting the lowest-order coefficient to s and taking random elements from K for the remaining $t - 1$ coefficients. [7]

The field K, the threshold t, the names $P_1, \ldots, P_n$ and the protocol below are known to all players. We assume that for each agent, there is a separate private communication channel with the dealer (for instance one based on public key encryption).

- *Distribution Phase*: The dealer has a secret $s \in K$, and selects a random polynomial $f(X) \in \Pi(t, s)$ and sends $s_i = f(P_i)$ as share in s privately to player P_i, $i = 1 \ldots n$.

[7] Note that this does not necessarily mean that one generates a polynomial of degree exactly $t - 1$, since $0 \in K$ is also in the play.

- *Reconstruction Phase*: From collection of $\geq t$ shares, the corresponding players pool their shares and jointly reconstruct $f(X)$ and compute $f(0) = s$.

By Lagrange interpolation it is clear that reconstruction works as desired. Indeed, if we are given the points (P_i, s_i), $i \in V$ for some set $V \subset \{1, \ldots, n\}$ and $|V| \geq t$, then the unique polynomial of degree smaller than $|V|$ that passes through these points, is of course $f(X)$.

Note that it is sufficient to compute $f(0) = s$, and that it is not necessary to reconstruct the full polynomial $f(X)$.

If we define the constants

$$\lambda_{V,i} := f_{V,i}(0),$$

then

$$s = f(0) = \sum_{i \in V} s_i \cdot \lambda_{V,i}.$$

In other words, *the secret is equal to a linear combination over the shares*, where the coefficients depends on the set V with $|V| \geq t$.

As to *privacy*, consider an arbitrary subset V of the agents of size $t-1$. Write $V = \{P_1', \ldots, P_{t-1}'\} \subset \{P_1, \ldots, P_n\}$, and write $s_1', \ldots, s_{t-1}'$ for the shares $f(P_1'), \ldots, f(P_{t-1}')$ of V.

Observe that for each $s' \in K$, the t points $(0, s')$, $(P_1', s_1'), \ldots, (P_{t-1}', s_{t-1}')$ uniquely determine a polynomial $f'(X) \in \Pi(t, s')$ that passes through all of them.

So from the joint view of the players in V, each secret is equally likely (take into account that the dealer chose f at random, given s) and hence the shares held by V give no information about the real secret s.

Note that since the joint view of any set of size $t-1$ gives no information about the secret, the view of a *smaller* subset doesn't give information about the secret either. This follows from the fact that a smaller subset holds even less information.

11 Verifiable Secret Sharing

In the presence of participants carrying out malicious attacks, there are two threats in Shamir's scheme.

- The dealer may send inconsistent shares, i.e. not all of them are simultaneously on some polynomial of degree smaller than t.
- At reconstruction, players may contribute false shares so that $\tilde{s} \neq s$ is reconstructed or nothing at all.

Note that if the malicious players coordinate well, the honest players cannot in general distinguish between "good shares" and "bad shares". Therefore, the honest players may not even be able to figure out who the malicious players are.

In this section we explain methods to remedy this situation.

11.1 Definition of Malicious Adversary

Before we define *verifiable secret sharing* (VSS) to remedy these threats, we make the model more precise and introduce some terminology. Consider a dealer and n agents. A malicious adversary is allowed to *corrupt* the dealer and any single subset of the n agents of size smaller than t. All other players are *honest*. Later during the execution of a protocol [8] the adversary is allowed to alter and control the behaviour of the corrupted players at his will, and even have them behave in a coordinated fashion. In particular the adversary can make some corrupted players crash. For simplicity, we assume that the adversary makes the choice of which subset to corrupt before anything happened, i.e. before the start of a protocol.

11.2 Definition of VSS

The following informal definition is based on a formal definition of VSS from [55].

1. If the dealer is honest, then the distribution of a secret s always succeeds, and the corrupted players gain no information about s as a result of the distribution phase. At reconstruction, the honest players recover s. These properties hold regardless of the behaviour of the corrupted players.
2. If the dealer is corrupt, then the following holds. Either the dealer is deemed corrupt by the honest players, and all of them abort the distribution phase. [9] Else, the distribution phase is accepted by the honest players and *some* value s is uniquely fixed by the information held by the honest players as a result of the distribution phase. In the reconstruction phase, the honest players recover this value s. These properties hold regardless of the behaviour of the corrupted players.

Note the absence of a secrecy condition in the corrupt dealer case: if the set of corrupted players includes the dealer, the adversary controlling them knows the secret. Therefore, it is only required that in this case the protocol is robust. The honest dealer case of course corresponds to what one would naturally require.

11.3 VSS Scheme

Here is a sketch of a generic construction of VSS based on a combination of Shamir's secret sharing scheme, commitments and zero-knowledge interactive proofs. Let the threshold t for Shamir's scheme satisfy $t - 1 < n/2$.

[8] In many multi-party computation protocols, the dealer will in fact at the same time also be one of the agents. In this case, there are in total n players involved, and the condition on the adversary is equivalent to saying that he is allowed to corrupt any single subset of size less than t of the n players, without distinguishing between dealer and agents.

[9] Another possibility is that the honest players take some default set of shares.

Commitments We assume that we have a commitment scheme for committing to $\log |K|$ bits, for instance obtained as a parallel version of the commitments from Section 6.2 based on RSA or trapdoor one-way permutations.

From a high level, a commitment protocol based on such primitives works as follows. There is a public, efficiently computable function "commit" whose description follows from the primitive chosen, and it takes as input a random m-bit string (for some m that will be clear from the primitive) and some $\log |K|$-bit string.

To commit to a $\log |K|$-bit string x, one chooses a random m-bit string ρ and computes $C = \text{commit}(x, \rho)$. Finally, one publishes C as a commitment.

To open, one publishes x and ρ. The opening is verified by checking that $\text{commit}(x, \rho) = C$. We call the string ρ the opening information of the commitment to x.

Broadcast We now also assume that a primitive called *broadcast* is at the disposal of the participants. This is a mechanism by means of which any of the participants can make sure that a message he has for all players is received unaltered by the honest players, despite the possible presence of malicious adversaries. Furthermore, we assume that recipients can establish who is the originator of the message. This mechanism may be realized by physical means or may be simulated by a protocol among the players. It suffices to know that it can be realized using digital signatures for instance.

VSS Protocol Here is an informal overview of a VSS protocol due to [59], which is also a nice illustration of the power of zero knowledge techniques and commitments. We assume that all players are restricted to probabilistic polynomial time computations.

- *Distribution Phase*: The dealer has secret $s \in K$, and computes shares $s_1, \ldots, s_n$ of s as in Shamir's scheme. For $i = 1 \ldots n$, he computes a commitment C_i to s_i. After he has broadcast the commitments to all players, he proves in zero knowledge to all players $P_1, \ldots, P_n$ that the commitments contain shares consistent with some secret. If this proof is accepted, he sends s_i and the opening information for C_i privately to P_i, $i = 1 \ldots n$.
- *Reconstruction Phase*: As in Shamir's scheme, except that each player P_i not only broadcasts his share s_i, but also the opening information for C_i. For reconstruction of the secret s, the honest players only take those shares whose corresponding commitment is opened successfully.

We briefly analyze this protocol. Regarding the zero knowledge proof of consistency, we assume that it proceeds in such a way that consistency holds if and only if the proof is accepted by all honest players (except with negligible error of course).

There is a number of ways to achieve this, for instance by having each player separately and publicly (using the broadcast primitive) act as a verifier in a zero knowledge proof by the dealer, while all others verify whether the proof is accepting. Only if the dealer at some point returns a proof that is not accepting, the honest players accuse the dealer and abort. Note that if consistency does not hold, then with high probability the proof when verified by an honest player will fail.

The actual consistency statement that the dealer has to prove, could take the following form: there exist $s, a_1, \ldots, a_{t-1} \in K$, $\rho_1, \ldots, \rho_n \in \{0,1\}^m$ such that for $i = 1 \ldots n$, $C_i = \text{commit}(s + \sum_{1 \leq j \leq t-1} a_j P_i^j, \rho_i)$.

Such statements can be proved in zero knowledge by the methods of [58], for instance. [10]

If the dealer is honest, the distribution phase definitely succeeds. Privacy follows from the hiding property of the commitments (the corrupted players are polynomially bounded and hence cannot read the contents of commitments), the privacy of Shamir's scheme, and the zero knowledge property of the proofs.

Looking at the reconstruction phase, and assuming that the distribution phase was successful, we note that in the case of corrupt shares, the commitments cannot be successfully opened since this would contradict the binding property. Therefore, false shares are always found out and can henceforth be ignored by the honest players. In summary, the only malicious action the corrupt players can undertake is to refuse to participate in the reconstruction phase. But since there are at most $t-1$ corrupt players and since we assumed that the threshold t in Shamir's scheme satisfies $t - 1 < n/2$, there are always t honest players [11] to reconstruct the secret s.

11.4 Other Work

Particularly efficient VSS based on specific intractability assumptions (discrete logarithms) are presented in [46] and [72]. See also [29]. In a later section we discuss information theoretic VSS.

12 GMW: Achieving Robustness

With VSS in hand, the GMW protocol first has each player VSS each of his inputs before the n-player extension of protocol from Section 6.2 is executed (see Section 9). At the end of the protocol, each player applies VSS again, this time to the (additive) shares in the result of the computation. This requires additional zero knowledge proofs (in the same style as before) showing that these additive shares are indeed shared with VSS.

If one of the players fails in this phase (or earlier) he is kicked out of the computation, and the remaining players back up to the beginning, reconstruct the failed player's input, and do the protocol over again, this time simulating

[10] A particularly efficient general zero knowledge protocol is given in [33]

[11] This argument can also be used to show that $t - 1 < n/2$ is optimal, i.e. it is not only sufficient but also necessary.

the failed player openly. Note that up to $t-1$ corrupted parties are tolerated in this way. With a similar argument as in the case of VSS, this can be shown to be optimal. There are more efficient variants, see [62] for a full description and analysis of the GMW-result.

The analysis [62] of the actual [12] GMW-protocol is very complex and has to deal with many subtleties that have been suppressed in our informal overview.

13 Other Work

Chaum, Damgaard and van de Graaf [23] present protocols where one of the players' input is unconditionally protected. Kilian, Micali, and Ostrovsky [68] show how oblivious transfers can be used in zero knowledge protocols. Galil, Haber and Yung [53] achieve greater efficiency with their computation protocols. Recently, Gennaro, Rabin and Rabin [57] presented particularly efficient protocols for the cryptographic model (see also [29]).

[12] We have made a number of simplifications for ease of exposition. For instance, we have neglected *input independence.*: in reality one must make sure that the corrupted parties choose their inputs to the computation independently from the inputs of the honest parties. This can be achieved by having all players commit to their inputs and having them give a zero knowledge proof of knowledge showing they can open these commitments.

Part III

Information Theoretic Security

14 Introduction

In 1987, two independent papers by M. Ben-Or, S. Goldwasser and A. Wigderson [13], and D. Chaum, C. Crépeau and I. Damgaard [24] achieved a new breakthrough in the theory of general multi-party computations.

They demonstrated the existence of *information theoretically secure* general multi-party computation protocols.

The price to be paid is a smaller tolerance with respect to the number of maliciously behaving players. Whereas [59] tolerates any malicious minority under the assumption that the players are computationally bounded, the protocols of [13] and [24] tolerate any malicious subset of size less than a third of the total number of players, with no assumptions on the computational power of the adversary. However, both papers argue that this is essentially the best one can achieve.

A common feature of both papers is the use of Shamir's secret sharing scheme, and the general paradigm of compiling a protocol secure against semi-honest players into one secure against malicious players by forcing all players to prove that they behave as semi-honest ones. However, [13] relies on techniques from the theory of error correcting codes, while [24] is based on *distributed* commitments and zero-knowledge. The result from [13] achieves perfect correctness, while [24] has a negligibly small error probability. [13]

14.1 Model

We make the model of BGW [13] and CCD [24] a bit more precise.

Communication:

- The n players are arranged in a complete (synchronous) network.
- Untappable private channels between each pair of players are available.

Adversary:

- The adversary is allowed to corrupt any single subset of size k of the players before the start of the protocol.
- Exercising complete control over the corrupted players, the adversary is allowed to force the corrupted players into coordinated malicious attack on the protocol.

Function:

- Any efficiently computable function g with n inputs.

[13] An interesting side-contribution of [24], seemingly often overlooked, is that it employs general zero knowledge techniques and information theoretically secure commitments in a distributed setting, showing that general zero knowledge makes sense (in a distributed setting) even if for instance $NP = P$.

14.2 Results of CCD and BGW

There is a **correct**, **private**, **robust** polynomial time protocol evaluating g iff the adversary corrupts at most $k < n/3$ players. In the semi-honest case this is $k < n/2$. These bounds are optimal. [14]

14.3 Remark on Broadcast

For security against malicious attacks, both results require the availability of a broadcast channel [69]. It is clearly not an option to use digital signatures in this case, since this does not fit with context of information theoretically secure protocols.

However, broadcast among n players can be efficiently simulated even in the presence of at most $t-1 < n/3$ malicious players (see for instance [47, 54]). This bound is optimal.

14.4 Outline of this Part

Instead of explaining the techniques of [24] and [13], we will sketch proofs of their results based on recent developments in the theory of multi-party computation due to Gennaro, Rabin and Rabin [57] and Cramer, Damgård and Maurer [29].

We first treat the semi-honest case.

15 Semi-Honest Case

We show how n players can securely compute on shared secrets. More concretely, n players have shares in two secrets (according to Shamir's secret sharing scheme) and they wish to compute from these, random shares in the sum or the product of these secrets.

We first treat these two cases. Later we show how this allows the n players to compute an arbitrary function on shared secrets.

15.1 Computing on Shared Secrets

Constants, Addition Suppose there are n players holding shares of two secrets s and s', both resulting from Shamir's secret sharing scheme, with parameters n and t. It is easy to see that shares for the sum $s + s'$ are obtained when each player simply adds his shares of s and s'. Moreover, if the players later reconstruct $s+s'$ from these new shares, no new information about s, s' is given away beyond what can be deduced from their sum $s + s'$.

[14] Given a broadcast channel for free, a malicious minority can also be tolerated by the result of T. Rabin and M. Ben-Or [75, 74] at the expense of a negligble correctness error.

If the dealer used polynomials f and g to compute the shares of s and s' respectively, the new shares "look" as if the dealer had used the polynomial $(f+g)$ to compute shares of $s+s'$.

Similarly, for any constant c *known* to all players, they compute shares for $c \cdot s$ (respectively, $c+s$) by multiplying (respectively, adding) each share by c.

Multiplication We explain a method introduced by R. Gennaro, T. Rabin and M. Rabin [57]. Suppose there are n players holding shares of two secrets s and s', both resulting from Shamir's secret sharing scheme, with parameters n and t.

The goal of the players is to jointly compute on their shares such that as a result they hold shares in the product $s \cdot s'$, also resulting from Shamir's secret sharing scheme with parameters n and t. Moreover, they require that these shares for $s \cdot s'$ are randomly generated, as if the (honest) dealer had not only distributed shares for s and s', but independently for $s \cdot s'$ as well.

If we consider the joint view of any set of $t-1$ players, we can observe that this randomness condition has the following effect. If $s \cdot s'$ is later reconstructed from the n shares of $s \cdot s'$, the shares revealed together with the complete information held by the $t-1$ players, do not give information beyond ss' and what can be inferred from it.

Unlike the case of addition, this is not trivial to solve. The first protocols for this task appeared in [24] and [13], but the solution of [57] we explain here is elegant and simple.

Let f and g denote the polynomials used by the dealer. Let n denote the number of players (agents) and t the threshold. We assume that $t-1 < n/2$.

We have: $f(0) = s$ and $g(0) = s'$ and both polynomials are of degree less than t. For $i = 1 \dots n$, write $s_i = f(P_i)$ and $s'_i = g(P_i)$ for player P_i's shares in secrets s and s', respectively.

We are interested in $s \cdot s'$. Observe that the polynomial $f \cdot g$ satisfies $(f \cdot g)(0) = s \cdot s'$ and that it has degree at most $2t-2 < n$. Furthermore, for $i = 1 \dots n$, $(f \cdot g)(P_i) = s_i \cdot s'_i$, which is a value that player P_i can compute on his own.

Therefore, by Lagrange interpolation and by our assumption $t-1 < n/2$, the players at least hold enough information to define $f \cdot g$ uniquely.

Now comes the interesting point. First, there exists a fixed linear combination, whose coefficients $r_1, \dots, r_n \in K$ only depend on the P_i's, over the "product-shares" $s_i \cdot s'_i$ that yields $s \cdot s'$. This is easy to see. By the Lagrange interpolation formula we have

$$(f \cdot g)(0) = \sum_{1 \le i \le n} \left(\frac{\prod_{1 \le j \le n, j \ne i} -P_j}{\prod_{1 \le j \le n, j \ne i} (P_i - P_j)} \right) \cdot s_i s'_i.$$

So the values between brackets are the coefficients $r_1, \dots, r_n$, and all of these can be computed from public information. A simple but important fact for the analysis to follow is that at least t of these values are non-zero. For suppose without loss of generality that $r_t = \dots = r_n = 0$. Then we have, for instance,

$(f \cdot \mathbf{1})(0) = s = \sum_{1 \le i \le t-1} r_i (s_i \cdot 1)$, where $\mathbf{1}$ denotes the polynomial $1 \in K[X]$. This would mean that players $P_1, \ldots, P_{t-1}$ can break Shamir's secret sharing scheme, a contradiction.

Of course the players don't want to keep these product-shares $s_i s'_i$ as their shares in $s \cdot s'$; first of all it changes the threshold, and second, these product-shares are by no means random shares of the secret $s \cdot s'$. In fact, reconstruction could reveal more than just $s \cdot s'$ in the sense that it could also reveal information about s, s' individually.

Therefore, they first *re-share* their product-shares: each player P_i acts as a dealer and distributes shares of his secret $s_i \cdot s'_i$ to all players (P_i included for completeness), using the same parameters n and t and resulting in share u_{ij} for player P_j, $j = 1 \ldots n$. Write h_i for the polynomial used by P_i.

Consider the polynomial

$$h(X) = \sum_{1 \le i \le n} r_i \cdot h_i(X).$$

This has degree $< t$, and $h(0) = \sum_{1 \le i \le n} r_i \cdot h_i(0) = \sum_{1 \le i \le n} r_i \cdot s_i s'_i = s \cdot s'$.

Therefore, when each player P_i now computes

$$v_i = \sum_{1 \le j \le n} r_j u_{ji},$$

P_i has a share v_i in $s \cdot s'$ resulting from the polynomial $h(X)$ of degree less than t.

As to privacy, it is sufficient to note that from the point of view of any coalition of the players of size $t - 1$ or smaller, the polynomial h contains at least one h_i contributed by a player outside the coalition, since at least t of the $r_1, \ldots, r_n$ are non-zero.

15.2 Protocol for Semi-Honest Participants

Based on the techniques for computing on shared secrets, we now present a general multi-party computation protocol (essentially due to [57]) secure if a semi-honest adversary has access to the complete information of at most $t - 1$ players, where $t - 1 < n/2$.

We assume that the function they wish to jointly compute is given as an arithmetic circuit over a finite field K with $|K| > n$. Arithmetic circuits are similar to Boolean circuits (see Section 4), except that the computations take place over K instead of $GF(2)$. This is no restriction: if we fix an arbitrary K, then any function that is efficiently computable is also computable by a polynomial size arithmetic circuit over K. These are the types of gates we require: two-input addition- and multiplication gates, and one-input gates for addition or multiplication with a constant.

As in Section 4, the computation proceeds in a gate-by-gate manner, maintaining the invariant that at each point the players have random shares in the current intermediate results.

When they have processed the final output gate, all players broadcast their shares in the result, and reconstruct it.

Input Distribution Phase
Using Shamir's Secret Sharing Scheme, each player provides shares of his input to all players.

Computation Phase
If the current gate is addition, or addition/multiplication of a constant, they follow the steps from the first part of Section 15.1. If the current gate is multiplication, they follow the steps from the second part Section 15.1.

Reconstruction Phase
Each player broadcasts his share in the output, and all reconstruct the result.

15.3 Optimality of the Bound

Suppose there exists an integer $n > 1$ and a general n-party computation protocol secure if more than a strict minority of the players conspire (semi-honestly), i.e. the number of tolerable conspirators would be at least $n/2$. This would immediately imply a protocol for two players to evaluate for instance the AND-function obliviously (each of the players would simulate a different half of the n players). By the same arguments as in Section 3.1, this is impossible and hence the $t - 1 < n/2$ bound is optimal.

16 Verifiable Secret Sharing

We first show how to turn Shamir's secret sharing scheme into a Verifiable Secret Sharing Scheme. Based on this, we construct distributed homomorphic commitments. Finally, we explain how to defend against malicious attacks in general multi-party computations.

These results are taken from Cramer, Damgaard and Maurer [29].

16.1 Linear Algebraic View on Shamir's Secret Sharing

We adopt a linear algebraic view on Shamir's secret sharing scheme, that some may find less intuitive than the explanation based on polynomial interpolation (though technically speaking it is definitely as elementary).

Our reasons for doing so are two-fold.

First, it opens the way to a verifiable secret sharing scheme that avoids the bi-variate polynomials and error correcting codes of [13].

Second, Brickell [18] points out how this linear algebraic view leads to a natural extension to a wider class of secret sharing schemes that are not necessarily of the threshold type. This has later been generalized to all possible so-called monotone access structures [15] Karchmer and Wigderson [66] based on a linear algebraic computational device called *monotone span program.*

Cramer, Damgård and Maurer [29] extend these results of Karchmer and Wigderson, by introducing a method to transform monotone span program based secret sharing schemes (Shamir's scheme is a particular instance) into verifiable secret sharing schemes. The enhancement is purely linear algebraic in nature and admits no analogous view based on polynomials. In fact, in the monotone span program model of [66], which deals with arbitrary monotone access structures and not just threshold ones, it is in general not possible to speak about polynomials. Therefore, one reaches further if one concentrates on the quintessential algebraic properties, instead of on the very specific language of polynomials.

We will not present the general VSS result of [29] here, but rather the threshold-case which has some nice extras over the general construction, that are mentioned but not detailed in [29]. The presentation is self-contained and doesn't require knowledge of [66].

Let K be a finite field, let M be a matrix with n rows and t columns, and with entries from K. We say that M is an (n,t)-Vandermonde matrix (over K) if there are $\alpha_1,\ldots,\alpha_n \in K$, all distinct and non-zero, such that the i-th row of M is of the form $(1,\alpha_i,\ldots,\alpha_i^{t-1})$ for $i=1\ldots n$. Note that this implies that $|K|>n$.

For an arbitrary matrix M over K with n rows labelled $1\ldots n$, and for an arbitrary non-empty subset A of $\{1,\ldots,n\}$, let M_A denote the matrix obtained by keeping only those rows i with $i \in A$. If $A=\{i\}$, we write M_i. Similarly, for a vector $\mathbf{s} \in K^n$, $\mathbf{s}_A$ denotes those coordinates s_i of $\mathbf{s}$ with $i \in A$.

Let M_A^T denote the transpose of M_A and let $\mathrm{Im} M_A^T$ denote the K-linear span of the rows of M_A. We use $\mathrm{Ker} M_A$ to denote all linear combinations of the columns of M_A leading to $\mathbf{0}$, the *kernel* of M_A.

It is well-known that any square (i.e. number of rows is equal to number of columns) Vandermonde matrix has a non-zero determinant. If M is an (n,t)-Vandermonde matrix over K and $A \subset \{1,\ldots,n\}$, then we conclude that the rank of M_A is maximal (i.e. is equal to t, or equivalently, $\mathrm{Im} M_A^T = K^t$) if and only if $|A| \geq t$.

But more is true. Let $\boldsymbol{\epsilon}$ denote the column vector $(1,0,\ldots,0) \in K^t$. If $|A|<t$, then $\boldsymbol{\epsilon} \notin \mathrm{Im} M_A^T$, i.e. there is no $\boldsymbol{\lambda} \in K^{|A|}$ such that $M_A^T \boldsymbol{\lambda} = \boldsymbol{\epsilon}$.

This can be seen as follows. Suppose without loss of generality that $|A|=t-1$ and that there is such $\boldsymbol{\lambda}$. Consider the square matrix N_A obtained from M_A by deleting the first column (that consists of $t-1$ 1's). This matrix is "almost" a square Vandermonde matrix: it can be seen as a square Vandermonde matrix multiplied by a matrix that has zeroes everywhere, except that its diagonal consists of non-zero elements (in fact, α_i's with $i \in A$). It follows that N_A has a

[15] This generalization has first been achieved by Ito, Nishizeki and Saito [65] and later by Benaloh and Leichter [11]. Both these results are based on elementary monotone formula complexity of the access structure ([65] is more restricted since it requires DNF formulae). However, the model of [66] is much more powerful in terms of efficiency. See also [14].

non-zero determinant. But then $M_A^T \boldsymbol{\lambda} = \boldsymbol{\epsilon}$ implies $N_A^T \boldsymbol{\lambda} = \mathbf{0}$ and $\boldsymbol{\lambda} \neq \mathbf{0}$. This is impossible since N_A is a square matrix with non-zero determinant.

Therefore we can say

$$\boldsymbol{\epsilon} \in \mathrm{Im} M_A^T \text{ if and only if } |A| \geq t.$$

We need some more basic linear algebra. For vectors $\mathbf{x}, \mathbf{y} \in K^t$, define the standard in-product (finite field case) as $\langle \mathbf{x}, \mathbf{y} \rangle = x_0 y_0 + \ldots + x_{t-1} y_{t-1}$. We write $\mathbf{x} \perp \mathbf{y}$ when $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ and $\mathbf{x}$ is said to be orthogonal to $\mathbf{y}$, and vice-versa. For a K-linear subspace V of K^t, $V^\perp$ denotes the collection of elements of K^t that are orthogonal to all of V (the *orthogonal complement*), which is again a K-linear subspace.

For all subspaces V of K^t we have $V = (V^\perp)^\perp$. This is an elementary fact that can be proved in a number of ways. Here we exploit the fact that K is finite.

Say $\dim(V) = t'$, and choose any basis for V. Now $\mathbf{x} \in V^\perp$ if and only if $\langle \mathbf{x}, \mathbf{f} \rangle = 0$ for all vectors $\mathbf{f}$ in the chosen basis. So if we arrange those basis vectors as the rows of a matrix M (it follows that $V = \mathrm{Im} M^T$), we have $V^\perp = (\mathrm{Im} M^T)^\perp = \mathrm{Ker} M$. The latter equality simply follows by inspection.

By Gaussian Elimination ("sweeping") on the rows of M, we can bring it of course into a form where the first t' columns constitute the identity matrix.

The rows of this new M are still a basis for V, and therefore the relationships above still hold. We count the number of $\mathbf{x}$ such that $M\mathbf{x} = \mathbf{0}$, i.e. we count $|\mathrm{Ker} M|$. From M's form, it follows that for each selection of the last $t - t'$ coordinates of $\mathbf{x}$, there is a unique selection of its first t coordinates such that $M\mathbf{x} = \mathbf{0}$. Hence, $|V^\perp| = |\mathrm{Ker} M| = |K|^{t-t'}$. Therefore, by applying this fact once more, $|(V^\perp)^\perp| = |K|^{t'}$. Since $V \subset (V^\perp)^\perp$ from the definition, it now follows that $V = (V^\perp)^\perp$.

By application of this fact, it now follows that $\mathrm{Im} M_A^T = (\mathrm{Ker} M_A)^\perp$, and we can conclude that

$$\boldsymbol{\epsilon} \notin \mathrm{Im} M_A^T \text{ if and only if there exists } \boldsymbol{\kappa} \in K^t \text{ such that } M_A \boldsymbol{\kappa} = \mathbf{0} \text{ and } \kappa_1 = 1.$$

Another simple identity is that $\langle \mathbf{x}, M_A^T \mathbf{y} \rangle = \langle M_A \mathbf{x}, \mathbf{y} \rangle$ for all $\mathbf{x}, \mathbf{y}$ of adequate dimensions.

Now we can present and analyze Shamir's scheme in an alternative fashion.

Let there be n players, and let t be the threshold. Over a finite field K, let M be an (n, t)-Vandermonde matrix.

Distribution Phase: Let $s \in K$ be the secret. The dealer chooses a vector $\mathbf{b} \in K^t$ by setting its first coordinate b_1 to s, and selecting random elements from K for the remaining coordinates. To player i he privately sends $s_i = M_i \mathbf{b}$ as share in s, for $i = 1 \ldots n$.

Reconstruction Phase: Let $A \subset \{1, \ldots, n\}$ with $|A| \geq t$. From their joint information, the players in A efficiently compute by elementary linear algebra $\boldsymbol{\lambda} \in K^{|A|}$ such that $M_A^T \boldsymbol{\lambda} = \boldsymbol{\epsilon}$. Write $M\mathbf{b} = \mathbf{s}$. Then $s = \langle \mathbf{b}, \boldsymbol{\epsilon} \rangle = \langle \mathbf{b}, M_A^T \boldsymbol{\lambda} \rangle = \langle M_A \mathbf{b}, \boldsymbol{\lambda} \rangle = \langle \mathbf{s}_A, \boldsymbol{\lambda} \rangle$, which they can compute efficiently.

It should be clear that reconstruction works as desired. Regarding privacy, let $|A| = t-1$, and consider the joint information held by the players in A, i.e. $\mathbf{s}_A = M_A\mathbf{b}$. Let $\tilde{s} \in K$ be arbitrary, and let $\boldsymbol{\kappa}$ be such that $M_A\boldsymbol{\kappa} = \mathbf{0}$ and $\kappa_1 = 1$. Then $M_A(\mathbf{b} + (\tilde{s} - s)\boldsymbol{\kappa}) = \mathbf{s}_A$ and the first coordinate of the argument is equal to $\tilde{s}$. This means that, from the point of view of the players in A, $\mathbf{s}_A$ can be consistent with the secret $\tilde{s}$.

The number of $\tilde{b} \in K^t$ with $\tilde{b}_1 = \tilde{s}$ is clearly equal to $|\mathrm{Ker}(M_A)|$ (which is independent of $\tilde{s}$), and the players in A have no information about s (take into account that all coordinates of $\mathbf{b}$ except possibly the first have been chosen at random).

16.2 Towards VSS

Let $t-1 < n/3$. A fact that is also exploited in [13] is that a complete set of shares $\mathbf{s}$ with at most $t-1$ arbitrary errors still defines the secret s uniquely.

Indeed, if this were not the case, then there would exist error vectors $\boldsymbol{\delta}_1, \boldsymbol{\delta}_2 \in K^n$ with Hamming-weight at most $t-1$, and vectors $\mathbf{b}_1, \mathbf{b}_2 \in K^n$ with distinct first coordinates b_1 and b_2, such that

$$\tilde{\mathbf{s}}_1 = \tilde{\mathbf{s}}_2,$$

where $\tilde{\mathbf{s}}_i = \boldsymbol{\delta}_i + M\mathbf{b}_i$, $i = 1, 2$.

However, there are at least $n-2(t-1) \geq t$ coordinates that are simultaneously zero in both error vectors. Hence, at least t coordinates of $\tilde{\mathbf{s}}_1 = \tilde{\mathbf{s}}_2$ are equal to the corresponding coordinates in $M\mathbf{b}_1$ and $M\mathbf{b}_1$. This means that $M\mathbf{b}_1$ and $M\mathbf{b}_1$ agree in at least t coordinates, and hence we must have $b_1 = b_2$. Note that the linearity of the secret sharing scheme is immaterial in this argumentation.

From the above we conclude that in principle and assuming that the dealer is honest, setting $t-1 < n/3$ guarantees robustness against players handing in false shares. However, efficiency is a problem (even when assuming an honest dealer): how to decode a "disturbed" set of shares $\tilde{\mathbf{s}}$ and recover the secret. In [13], efficient standard error correction techniques are applied to a version of Shamir's scheme obtained by first passing to an extension field of K.

We first explain how this can be avoided (for the moment we still assume an honest dealer).

Consider the following variant of Shamir's scheme.

Distribution Phase: Let $s \in K$ be the secret. The dealer chooses a random *symmetric* matrix $R \in K^{t,t}$, subject to the condition that it has s in its upper left corner. For $i = 1 \ldots n$, the dealer sends privately to player i the (row-)vector $\mathbf{s}_i = M_i R$ as share in s. Write $\mathbf{b}$ for the first column of R, then the first coordinate of $\mathbf{s}_i$ is equal to $M_i\mathbf{b}$. This value is called player i's *actual share* in s.

Reconstruction Phase: For $i = 1 \ldots n$, each player i broadcasts his share $\tilde{\mathbf{s}}_i$. Consider the matrix C with n rows and n columns, whose entry in the i-th row and j-th column is 1 if and only if $M_j\tilde{\mathbf{s}}_i^T = \tilde{\mathbf{s}}_j M_i^T$. Throw away all rows of C that have t or more zeroes. There will be at least t rows left. For each of the rows i left, take the first coordinate of the corresponding $\tilde{\mathbf{s}}_i$ as the actual share of player i. These at least t actual shares determine uniquely the secret s, according to Shamir's secret sharing scheme as before.

We first argue that the secret s is indeed efficiently reconstructed, assuming an honest dealer and at most $t-1$ arbitrarily corrupt players. First note that for all i, $(M_i R)^T = RM_i^T$ by symmetry of R. Hence, for all i, j we have

$$M_j \mathbf{s}_i^T = \mathbf{s}_j M_i^T.$$

From this we conclude that each player j holds a share $M_j \mathbf{s}_i^T$ in player i's actual share of s. Consider the case that a player i broadcasts a vector $\tilde{\mathbf{s}}_i$ that differs from his share $\mathbf{s}_i$ in the first coordinate (and possibly elsewhere as well), then for at most $t-1$ of the real $\mathbf{s}_j$'s we have $M_j \tilde{\mathbf{s}}_i^T = \mathbf{s}_j M_i^T$, since obviously, two complete sets of shares in Shamir's scheme (with parameters n, t) for different secrets can agree on at most $t-1$ of the shares. But we also have to take into account that not only player i may be cheating, he may also coordinate with $t-2$ other cheaters. Hence, an upperbound on the number of consistencies in this case is $(t-1)+(t-1) = 2t-2$. Therefore, there are at least t inconsistencies in this case.

On the other hand, if player i is honest then no matter how the corrupt players lie and cheat, they are going to cause at most $t-1$ inconsistencies in the i-th row of the consistency matrix C. Therefore, the procedure yields at least t good actual shares, sufficient for reconstructing s. Note that in the analysis we have only used the fact that the total information $\mathbf{s}_B$ received by the set of the honest players B is of the form $\mathbf{s}_B = M_B R$ for *some* symmetric R.

As to *privacy* we note the following. For vectors $\mathbf{v} = (v_1, \ldots, v_t) \in K^t$ and $\mathbf{w} = (w_1, \ldots, w_t) \in K^t$, the standard tensor product (matrix form)$\mathbf{v} \otimes \mathbf{w}$ is defined as a matrix with t rows and t columns such that the j-th column is equal to $v_j \mathbf{w}$. Note that $\mathbf{v} \otimes \mathbf{v}$ is a symmetric matrix. Privacy is argued in a similar way as in the case of the linear algebraic explanation of Shamir's scheme. Let $|A| \leq t-1$, and let $\boldsymbol{\kappa}$ satisfy $M_A \boldsymbol{\kappa} = \mathbf{0}$ and $\kappa_1 = 1$. Then $\boldsymbol{\kappa} \otimes \boldsymbol{\kappa}$ is symmetric, has 1 in its upper left corner and satisfies $M_A(\boldsymbol{\kappa} \otimes \boldsymbol{\kappa}) = \mathbf{0}$. This is then used to show that for each possible secret, the number of symmetric matrices with that secret in its upper left corner and consistent with the joint information of A, is the same.

16.3 Pairwise Checking Protocol and VSS

We now drop the assumption that the dealer is honest, and build a "pair-wise checking protocol", where each pair of players exchange checking information, around the scheme above to obtain VSS. The pair-wise checking as such is quite similar to methods from e.g. [13] and [47].

Let B denote the set of honest players, and let S_B be the total information received by B in the distribution phase. By the analysis of the honest dealer case above, we are done if $S_B = M_B R$ for *some* symmetric matrix R.

Suppose that some "pair-wise checking protocol" performed right after the dealer distributed the shares (as in the scheme above) would guarantee that

$$M_B S_B^T = S_B M_B^T.$$

We show that this is sufficient to conclude the existence of such R. Since certainly $|B| \geq t$, we know that the span of the rows of M_B is all of K^t. Hence there exists a matrix N_B such that $M_B^T N_B$ is equal to the identity matrix I_t with t rows and t columns. Hence we have

$$M_B(S_B^T N_B) = S_B,$$

and we can take

$$R = S_B^T N_B.$$

Note that this R is indeed symmetric, since from $M_B S_B^T = S_B M_B^T$ it follows that

$$N_B^T(M_B S_B^T)N_B = N_B^T(S_B M_B^T)N_B,$$

which implies that

$$I_t^T S_B^T N_B = N_B^T S_B I_t,$$

and hence

$$S_B^T N_B = N_B^T S_B = (S_B^T N_B)^T.$$

The following pairwise-checking protocol is appended to the distribution phase.

1. Each player i sends to each player j the value $M_j \mathbf{s}_i^T$. Player j checks that this is equal to $\mathbf{s}_j M_i^T$ (pairwise consistency check). In case of an inconsistency, player j broadcasts a complaint about the value received from player i.
2. In response to complaints, the dealer must broadcast the correct value $M_j \mathbf{s}_i^T$ for all complaints of players j about the values received from players i.
3. If any player j finds that the information broadcast by the dealer is still inconsistent, it is clear to player j that the dealer is corrupt, and he broadcasts a request that the dealer makes public all the information sent to player j. This counts as claiming that the dealer is corrupt. These accusing players remain passive until a decision is made in the final step.
4. The dealer must again broadcast all the requested information, and again this may result in some players accusing the dealer of being corrupt. This can repeat until the information broadcast by the dealer contradicts itself, or he has been accused by at least t players. Or else, no new complaints occur and the number of accusations is at most $t-1$. The decision whether or not to accept the distribution phase is now taken as follows. In the first two cases, the dealer is deemed corrupt and is disqualified. In the last case, the distribution phase is accepted by the honest players. Accusing players accept the information broadcast for them as their shares.

We analyze the protocol. First, we look at the honest dealer case. The corrupt players do not get more information than in the protocol above that assumes an honest dealer (note that no honest player will request the honest dealer to make public the information sent to him by the dealer, because if the honest player complains about some player, the honest dealer will always send the correct value).

Furthermore, the corrupt players can cause at most $t-1$ accusations, and hence the distribution phase is always accepted by the honest players if the dealer is honest.

Next, let's drop the assumption that the dealer is honest and let's assume that the distribution phase was accepted by the honest players. Then it is immediate that each honest player has a share that is consistent with the shares of all other honest players. Suppose that this is not the case. There must be at least one honest player that did not accuse the dealer (since there are at most $t-1$ accusations and at most $t-1$ corrupted players, and $2t-2<n$ since $t-1<n/3$). Clearly, the shares held by the set of non-accusing honest players (which is non-empty by the above) must be pair-wise consistent. All other shares of honest players are broadcast, so if there were any inconsistency, a non-accusing honest player would have accused the dealer, which is in contradiction with our assumptions.

17 General Protocol Secure against Malicious Attacks

Consider the protocol for the semi-honest case. We would like to enhance it so that the following invariant is maintained. At each point in the (once again) gate-by-gate multi-party computation, the current intermediate results (i.e. the values at the current gate as propagated through the circuit from the actual inputs) are secret shared (as in the semi-honest case) and moreover, each player is *committed* to his shares.

17.1 Homomorphic Distributed Commitments

Distributed commitments have similar binding and hiding properties as the commitments from Section 6.2, except that this time these properties hold unconditionally, i.e. regardless of the computing power of an adversary. Of course, this will be so only with respect to the adversary we have defined earlier, that corrupts less than $n/3$ of the players before the start of the protocol.

Based on VSS, this is how it works. If player j wants to commit to $s \in K$, the n players execute the distribution phase of VSS, where player j acts as the dealer and takes s as the secret. To open the commitment, the n players execute the reconstruction phase of VSS.

One can immediately see that given two distributed commitments to values s and s' respectively, a commitment to $s+s'$ is non-interactively created by having all players locally take the sum of the information they hold (i.e. the VSS-shares in s and s').

Similarly, they can take a commitment and non-interactively multiply or add in a known constant.

It is in this sense that we say that the commitments are homomorphic. To create a distributed commitment to ss', is more involved and is explained later on.

17.2 Maintaining the Invariant

Now think of the commitments from above as abstract, black-box homomorphic commitments, and forget for the moment how we actually constructed them. Consider a player that is committed to some secret s. We would like to force that player to share s correctly (according to Shamir's scheme) among all players such that each of the players is committed to his share.

To this end, the player acts as the dealer in Shamir's scheme, chooses random elements $\rho_1, \ldots, \rho_{t-1}$ and commits to them.

Then, by the homomorphic properties of the commitments and the fact that the shares in Shamir's scheme are linear combinations (with fixed public coefficients!) of the secret and the ρ_i's, the players can compute new commitments to these n shares by just doing local computations.

This guarantees that the dealer is committed to consistent shares, i.e. the shares results from a correct (not necessarily random, but this is no problem) execution of the distribution phase of the secret sharing scheme.

The only thing left to be done for each share is to open the corresponding commitment *privately* towards the recipient of that share, so that he actually learns the correct share.

In our particular case here, commitment is a VSS. Therefore, the latter opening consists of jointly executing the reconstruction phase of VSS, by sending all data from the VSS (=commitment) of a share privately to the owner of that share. Note that this always works, by the properties of our VSS, and regardless of whether the dealer cooperates or not.

We call this procedure *Commitment Sharing Protocol* (CSP)

In the *Input Distribution Phase* of the general protocol, all players will secret share their inputs to the computation in the way we have just described (CSP).

In the *Computation Phase*, if the current gate is addition, or multiplication by a constant, the procedure is trivial by the homomorphic properties of commitments and Shamir's secret sharing scheme. The only real difficulty left is handling multiplication gates, which we will study separately.

In the *Output Reconstruction Phase*, the commitments to the shares in the final result of the computation are opened. Each player collects enough correct shares to reconstruct the result (output) of the computation.

17.3 Linear Secret Sharing Schemes

We now set out to handle the multiplication gates. But first it is convenient to further explore our linear algebraic view.

Shamir's secret sharing scheme is a linear scheme in the sense that each share is a linear combination (with fixed, public constants) of the secret and random choices made by the dealer.

It is possible to take this point of view as the starting point for a class of secret sharing schemes [18, 14, 66]: *general linear secret sharing schemes.*

There are n players, and there is a public matrix M with d rows and e columns, in which each row is assigned to one of the players. Abstractly speaking, each of the d rows of M is labeled with exactly one element from $\{1, \ldots, n\}$, and we allow that some (or all) labels occur more than once. Write ψ for the function that associates a row with a player. For $A \subset \{1, \ldots, n\}$, let M_A denote those rows of M that are labeled with an element from the set A. If $A = \{i\}$, we write M_i.

To compute shares of a secret, the dealer chooses a vector $\mathbf{b}$ at random subject to the condition that the secret is in the first coordinate of the vector, and for $i = 1 \ldots n$ sends the vector

$$\mathbf{s}_i = M_i \mathbf{b}$$

as share in s privately to player i.

In Shamir's scheme this matrix corresponds of course to the Vandermonde matrix, and each player is associated with exactly one row.

Recall from the linear algebra proof of Shamir's scheme that exactly those subsets of the players can reconstruct the secret, whose matrix (i.e. the submatrix that contains the rows associated with the subset) has $\boldsymbol{\epsilon}$ in its K-linear span of the rows. Other subsets have no information about the secret.

It can be shown by similar arguments as the ones used in the linear algebra proof of Shamir's scheme, that in the general linear scheme as defined above, exactly those subsets A can reconstruct the secret for which $\boldsymbol{\epsilon}$ is in the K-linear span of the rows of M_A. Other subsets have no information.

Now in general, the subsets that can reconstruct are not exactly all subsets of a certain cardinality. One can show that for any *monotone access structure* Γ, i.e. a collection of subsets of the n players with the property that if A is a member of Γ than any set containing A is in Γ as well, there is a linear secret sharing scheme such that the subsets that can reconstruct the secret are exactly the members of Γ. Again, other subsets have no information.

Let $\boldsymbol{\epsilon}$ denote the vector $(1, 0, \ldots, 0) \in K^e$. It is not hard to show that the subsets A that can reconstruct the secret are exactly those for which $\boldsymbol{\epsilon} \in \mathrm{Im} M_A^T$. The players in such a set A jointly recover a secret s by computing

$$\langle \mathbf{s}_A, \boldsymbol{\lambda} \rangle = \langle M_A \mathbf{b}, \boldsymbol{\lambda} \rangle = \langle \mathbf{b}, M_A^T \boldsymbol{\lambda} \rangle = \langle \mathbf{b}, \boldsymbol{\epsilon} \rangle = s,$$

where $M_A^T \boldsymbol{\lambda} = \boldsymbol{\epsilon}$, and $\mathbf{s}_A$ are the shares held by A, i.e. $\mathbf{s}_A = M_A \mathbf{b}$.

The quadruple $\mathcal{M} = (K, M, \boldsymbol{\epsilon}, \psi)$ is called *monotone span program* [66]. This powerful device is said to compute an access structure Γ (or equivalently, a monotone Boolean function) if and only if it is the case that $\boldsymbol{\epsilon} \in \mathrm{Im} M_A^T$ exactly when A is a member of Γ.

We will also call the sets in this correponding access structure the sets "accepted" by $\mathcal{M}$. A set that is not accepted, is called "rejected".

So each linear secret sharing scheme can be viewed as derived from a monotone span program computing its access structure.

We now return to the multiplication protocol. Let M be a (n,t)-Vandermonde matrix over K with $t-1<n/2$. For vectors $\mathbf{s},\mathbf{s}'\in K^n$, define their *star-product*

$$\mathbf{s}*\mathbf{s}'=(s_1s_1',\dots,s_ns_n')\in K^n.$$

For vectors $\mathbf{x},\mathbf{y}\in K^t$, define their tensor product (this time a vector instead of a matrix)

$$\mathbf{x}\otimes\mathbf{y}=(x_1y_1,\dots,x_1y_t,\dots,x_ty_1,\dots,x_ty_t)\in K^{t^2}.$$

For a matrix M, let $M_\otimes$ denote M except that each row $\mathbf{v}$ of M is replaced by $\mathbf{v}\otimes\mathbf{v}$.

Another way to view the principle underlying the multiplication protocol from Section 15.1 for Shamir's scheme is by saying that there exists a fixed vector $\mathbf{r}\in K^n$, which we call *recombination vector*, such that for all $\mathbf{b},\mathbf{b}'\in K^t$, with respective first coordinates $s,s'\in K$, we have

$$\langle\mathbf{r},\mathbf{s}*\mathbf{s}'\rangle=ss',$$

where $\mathbf{s}=M\mathbf{b}$ and $\mathbf{s}'=M\mathbf{b}'$.

Call this the *multiplication-property* of the secret sharing scheme. The existence of the vector $\mathbf{r}$ follows for instance from the analysis in Section 15.1, as well as a method for efficiently computing it. From the analysis it also follows that Shamir's scheme has the multiplication property if and only if $t-1<n/2$.

In the case of defense against *malicious* attacks in the multiplication protocol for Shamir's scheme and for reasons to become clear shortly, we need additionally that for all $B\subset\{1,\dots,n\}$ with [16] $|B|\geq n-t+1$ there exists a fixed vector $\mathbf{r}$ (depending on B) such that

$$\langle\mathbf{r},\mathbf{s}_B*\mathbf{s}_B'\rangle=ss',$$

where $\mathbf{s}_B=M_B\mathbf{b}$ and $\mathbf{s}_B'=M_B\mathbf{b}'$ are arbitrary.

Call this the *strong multiplication-property* of the secret sharing scheme, and call $\mathbf{r}$ the *recombination vector* for the set B.

Note that if the strong multiplication-property is satisfied, then certainly also the multiplication-property is satisfied: just take $B=\{1,\dots,n\}$.

We can also say that strong multiplication is satisfied exactly when for each B with at least $n-t+1$ elements, M_B has multiplication.

If we now set $t-1<n/3$, then we see that for all B with $n-t+1$ elements, M_B is an $(n-t+1,t)$-Vandermonde matrix ("t out-of $n-t+1$") and also that $t-1<(n-t+1)/2$. If B has even more elements, this clearly holds as well. Therefore, strong multiplication is satisfied by the way we set the parameter t.

It will be helpful to further extend the linear algebraic view. Note that the definition of the multiplication-property makes no reference to Shamir's secret sharing or threshold access structures. We could require this property of a general linear secret sharing scheme. In fact, this is exactly the definition of *monotone*

[16] these sets of course correspond to the potentially honest sets rather than the potentially corrupt sets of size at most $t-1$

span programs with multiplication from [29]. For strong multiplication, the only change in the definition we make is to say that the property holds for all sets B that are the complement of a set that is rejected by the monotone span program (i.e. complements of sets that are not in the access structure).

It is proved [17] in [29] that $\mathcal{M} = (K, M, \boldsymbol{\epsilon}, \psi)$ is a monotone span program with multiplication if and only if

$$\boldsymbol{\epsilon} \otimes \boldsymbol{\epsilon} \in \mathrm{Im} M_{\otimes}^{T}.$$

Any vector $\mathbf{r}$ with $\boldsymbol{\epsilon} \otimes \boldsymbol{\epsilon} = M_{\otimes}^{T} \mathbf{r}$ is a recombination vector.

As to strong multiplication, let $\mathcal{M}_B$ be the monotone span program obtained by throwing away the rows corresponding to the complement B of a rejected set. Then it follows immediately that $\mathcal{M}$ has strong multiplication if and only if for all such B, $\mathcal{M}_B$ has multiplication.

We are now ready to state the properties we use in the explanation of defense against malicious attacks to follow. Now let $\mathcal{M}$ be a monotone span program with multiplication. We can now consider the linear secret sharing scheme based on $\mathcal{M}_{\otimes} = (K, M_{\otimes}, \boldsymbol{\epsilon} \otimes \boldsymbol{\epsilon}, \psi)$ and conclude that the set $\{1, \ldots, n\}$ is accepted by $\mathcal{M}_{\otimes}$. Hence, if the n players receive a complete set of shares $M_{\otimes}\mathbf{c}$, they can recover the secret, which is $\mathbf{c}$'s first coordinate. This follows from the observations about the connection between general linear secret sharing and monotone span programs above.

If $\mathcal{M}$ has strong multiplication, this is true for each subset B whose complement is rejected by $\mathcal{M}$. This fact and the following technicality (which is proved directly from the definitions) are useful in what follows.

For any monotone span program $\mathcal{M}$, and for all $\mathbf{b}$ and $\mathbf{b}'$, we have

$$\mathbf{s} * \mathbf{s}' = M_{\otimes}(\mathbf{b} \otimes \mathbf{b}'),$$

where $\mathbf{s} = M\mathbf{b}$ and $\mathbf{s}' = M\mathbf{b}'$.

17.4 The Commitment Multiplication Protocol

The situation is as follows. There are two values s and s', and each of the n players is committed to his shares in s and s'.

We'd like to have a protocol by means of which the same can be enforced on ss'.

Of course the protocol from Section 15.1 comes in handy, but we will have to enhance it.

Let $\mathcal{M} = (K, M, \boldsymbol{\epsilon}, \psi)$ be the monotone span program underlying Shamir's secret sharing scheme with $t - 1 < n/3$.

Consider player i right before he re-shares $s_i s_i'$ in Section 15.1, where s_i and s_i' are his shares in s and s', respectively. In the current context we may assume that he is already committed to s_i and s_i' separately.

[17] This follows from the definition and the uniqueness of algebraic normal form.

It is sufficient for our purposes here if player i could create a commitment to $s_i s_i'$ and convince the rest of the players that this is indeed a commitment to $s_i s_i'$.

Indeed, suppose we had such a method, then for re-sharing we would do as in Section 15.1 and additionally have each player i commit to his local product $s_i s_i'$, prove that the resulting commitment contains indeed $s_i s_i'$, and subsequently apply the Commitment Sharing Protocol (CSP) to it.

After each player i has done so, they can compute a CSP of ss' using the linearity of the CSP and using the recombination vector $\mathbf{r}$.

Therefore, it is only left to show how player i can prove that a given commitment contains the product of the contents of two other given commitments.

We assume that $t-1 < n/3$. Let M be an (n,t)-Vandermonde matrix. Then $\mathcal{M} = (K, M, \boldsymbol{\epsilon}, \psi)$ is with strong multiplication and ψ just associates the j-th row of M with the j-th player, $j = 1 \ldots, n$.

Player i is committed to s_i and s_i'. The then protocols starts by executing the CSP twice, once with s_i being the secret, and finally and independently with s_i' being the secret.

This results in (committed) shares

$$(u_1, \ldots, u_n) = \mathbf{u} = M\mathbf{b},$$

in the secret s_i, where (for $j = 1 \ldots n$) player j holds u_j, and (committed) shares

$$(u_1', \ldots, u_n') = \mathbf{u}' = M\mathbf{b}',$$

in the secret s_i', where (for $j = 1 \ldots n$) player j holds u_j'.

Player i proceeds by committing to $s_i s_i'$, and to each of the t^2 coordinates of $\mathbf{b} \otimes \mathbf{b}'$. Finally, CSP is executed with these choices made by player i, but this time on the linear secret sharing scheme defined by $M_\otimes$, instead of M. This results in (committed) shares (with respect to $M_\otimes$!)

$$\mathbf{v} = (v_1, \ldots, v_n) = M_\otimes(\mathbf{b} \otimes \mathbf{b}'),$$

in the secret $s_i s_i'$, where (for $j = 1 \ldots n$) player j holds v_j.

Note that if player i indeed committed to the correct value $s_i s_i'$ (and not to some value $\tilde{\sigma} \neq s_i s_i'$) and indeed committed to the coordinates of $\mathbf{b} \otimes \mathbf{b}'$, then

$$(u_1 u_1', \ldots, u_n u_n') = (v_1, \ldots, v_n),$$

since $\mathbf{u} * \mathbf{u}' = M_\otimes(\mathbf{b} \otimes \mathbf{b}')$.

Now assume that it is perhaps the case that player i did *not* commit to $s_i s_i'$. In any case, there is a vector $\mathbf{c}$ such that

$$(v_1, \ldots, v_n) = M_\otimes \mathbf{c},$$

by the properties of CSP. Here, c_1 is the actual (committed) value that is *supposed* to be equal to $s_i s_i'$.

We now assume that $c_1 \neq s_i s_i'$, and prove that this leads to an inconsistency with the information held by at least one honest player, and that he can prove that there is an inconsistency.

Write $\mathbf{u} * \mathbf{u}' = \mathbf{w}$. Consider the set B, defined as the complement of the set of players that the adversary actually corrupted (i.e. B consists of the honest players). Note that $|B| \geq n - t + 1$. Since $\mathbf{u} * \mathbf{u}' = M_{\otimes}(\mathbf{b} \otimes \mathbf{b}')$ and since $\mathcal{M}$ has strong multiplication, B is accepted by $\mathcal{M}_{\otimes}$ and the set of shares $\mathbf{w_B}$ for B defines $s_i s_i'$ uniquely. Likewise, $\mathbf{v}_B$ defines a secret (i.e. $\mathbf{c}$'s first coordinate c_1) uniquely.

Therefore, if $c_1 \neq s_i s_i'$, there must be a $j \in B$ such that player j holds different shares for c_1 and $s_i s_i'$: if not, the reconstruction procedure for B (in the secret sharing scheme derived from $\mathcal{M}_{\otimes}$) applied to $\mathbf{w}_B$ and $\mathbf{v}_B$ would yield the same secrets.

Therefore, if player i did not commit to $s_i s_i'$ there is at least one honest player j that will notice an inconsistency and is going to complain. Upon that complaint, the commitments to $u_j u_j'$ and v_j are opened so that all honest players conclude that player i is corrupt.

On the other hand, if player i is honest, then there are at most $t-1$ such complaints from the corrupted players, and each of them will not convince any honest player, since opening the commitments will show that the complaining player is corrupt rather than player i. Moreover, the information that becomes available in the course of handling these complaints, does not yield any new information (from the point of view of the corrupted players) about $s_i s_i'$.

17.5 Extensions

The protocol above [18] and its analysis are a special case of [29]. In fact, the basic framework behind it also works for any adversary that can be captured [19] by a monotone span program with (strong) multiplication.

However a lot of things have to be settled first. The VSS protocol we described is an optimization for the threshold case of the general VSS scheme from [29]. That scheme is based on arbitrary monotone span programs and we cannot in general assume as in the threshold case here, that the matrix corresponding to the honest players has maximal rank (this is essential in the analysis of the threshold VSS). However, one can show that the protocol, although in general not a VSS, is still a distributed commitment scheme. Based on these commitments, one can indeed construct VSS based on arbitrary monotone span programs.

[18] We have not tried to optimize its efficiency, and we have been not very explicit about how to handle situations where players are found out to be corrupt. In any case, it is always possible to back-up to the beginning, and recover the inputs of corrupted players, after which the protocol is done over again with the corrupted players openly being simulated. There are other options which we do not discuss here.

[19] Loosely speaking, this requires a monotone span program with (strong) multiplication that rejects the sets in the adversary structure: a pre-determined collection of subsets of the players, out of which the actual adversary may pick an element and corrupt all the players in it.

Moreover, [29] provides a theory of monotone span programs with (strong) multiplication that shows that exactly those general (not necessarily threshold) adversaries are captured for which [64] demonstrates that secure computation tolerating them is possible at all. Therefore, the theory is complete.

Upper bounds on the complexity of monotone span programs with (strong) multiplication are given as well, that show significant improvements over previous approaches (similarly for VSS, but not requiring multiplication properties).

It is proved [29, 30] that for all relevant monotone functions f (i.e. Q2-functions), if a monotone span program of size m is given that computes such a function f, then there exists a monotone span program *with multiplication* that computes f as well and has **size** $O(m)$. Note that the novelty is in the last part of the claim. The proof is included in Appendix A. This implies (see [29, 32]), in a well-defined sense, that *linear secret sharing is "sufficient" for general secure multi-party computation*, where both existence and efficiency are taken into account.[20].

A remark about broadcast is in place. In case of general adverarsaries, information theoretically secure broadcasts protocols defending against threshold adversaries are in general not sufficient. Therefore, [29] uses the result of [49].

Also, the techniques extend to the model of [75], where broadcast is assumed (and cannot be simulated information theoretically) and an exponentially small error is tolerated (see also [32]). This is non-trivial, and we omit any of the details.

18 Other Work

We provide some suggestions for further reading (besides those references already given). This list is by no means complete and selection has been quite ad-hoc (This holds as well for the results covered in detail in this paper, with the exception of the classical results).

Adaptive adversaries, i.e. adversaries who do not necessarily select their vicitms before the start of the protocol but rather adaptively as the protocol is proceeding, are dealt with in [8] [20].

In [2] it is shown how general multi-party computations can be performed with polynomial complexity and a constant number of rounds of interaction, provided that the function to be evaluated is given as a polynomial size arithmetic formula (instead of circuit). Efficiency considerations (also using pre-processing) are discussed in [5, 6].

This issue of a corrupt majority is studied in [3].

Secure multi-party computation in an asynchronous communication model is addressed in [12].

[20] In [31] it is shown that there is no efficient construction of general multi-party computation protocols from "black-box secret sharing". Hence, "structure", such as linearity, is a requirement. On the other hand, it is demonstrated in [31] that VSS protocols *can* be efficiently constructed from a black-box secret sharing scheme

Loosely speaking, a proactively secure protocol is one secure against an attacker who in principle can corrupt an arbitrary number of players in the life-time of a system, except that in each time-frame less than, say, half of the players are corrupted and a majority is honest [71, 50].

For lots of references and detailed explanations of some fundamental results, see for instance [51] and [19].

Regarding multi-party computation protocols for electronic cash or electronic voting, see for instance [22], [26], [35] and [34].

Threshold cryptography, i.e. efficient and secure distributed computation for specific functions was introduced in [42]. See for instance, [41], [56] and [73] for distributed RSA-protocols.

A solution for the problem of *non-interactive* computation on encrypted data in the two-party scenarion is given in [78].

19 Acknowledgements

Donald Beaver, Claude Crépeau, Ivan Damgård, Serge Fehr, Matthias Fitzi, Martin Hirt, Ueli Maurer, Săsa Radomirović, Berry Schoenmakers are kindly acknowledged for discussions, answering questions, giving comments or remarks. Finally, also thanks to the students of the ETH Advanced Cryptographic Protocols Course (1999) for providing comments on parts of the manuscript.

References

1. W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr: *RSA and Rabin functions: Certain parts are as hard as the whole*, SIAM Journal on Computing, 17(2):194-209, April 1988.
2. J. Bar-Ilan and D. Beaver: *Non-cryptographic fault-tolerant computing in constant number of rounds of interaction.* In Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, pages 201-209, Edmonton, Alberta, Canada, 14-16 August 1989.
3. D. Beaver and S. Goldwasser: *Multiparty computation with faulty majority (extended announcement)*, In 30th Annual Symposium on Foundations of Computer Science, pages 468-473, Research Triangle Park, North Carolina, 30 October-1 November 1989. IEEE
4. D. Beaver: *Foundations of Secure Interactive Computing*, Proceedings of Crypto 91, Springer Verlag LNCS, vol. 576, pp. 420-432, Springer-Verlag, 1992.
5. D. Beaver: *Secure Multiparty Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority*, J. Cryptology 4:2 (1991), 75-122.
6. D. Beaver: *Efficient Multiparty Protocols Using Circuit Randomization*, Proceedings of Crypto '91, Springer-Verlag LNCS, 1992, 420-432.
7. D. Beaver: *How to break a "secure" oblivious transfer protocol.*, Eurocrypt '92, volume 658 of Lecture Notes in Computer Science, pages 285-296. Springer-Verlag, 24-28 May 1992.
8. D. Beaver and S. Haber: *Cryptographic protocols provably secure against dynamic adversaries*, volume 658 of Lecture Notes in Computer Science, pages 307-323, Springer-Verlag, 24-28 May 1992.

9. D. Beaver: *Equivocable Oblivious Transfer*, Proceedings of Eurocrypt '96, Springer–Verlag LNCS 1070, 1996, 119–130.
10. D. Beaver: *Adaptively Secure Oblivious Transfer*, to appear in the Proceedings of Asiacrypt '98.
11. J. Benaloh, J. Leichter: *Generalized Secret Sharing and Monotone Functions*, Proc. of Crypto '88, Springer Verlag LNCS series, pp. 25–35.
12. M. Ben-Or, R. Canetti, O. Goldreich: *Asynchronous Secure Computations*, Proc. ACM STOC '93, pp. 52–61.
13. M. Ben-Or, S. Goldwasser, A. Wigderson: *Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, Proc. ACM STOC '88, pp. 1–10.
14. M. Bertilsson, I. Ingemarsson: *A Construction of Practical Secret Sharing Schemes using Linear Block Codes*, Proc. AUSCRYPT '92, LNCS 718 (1993), 67-79.
15. G. R. Blakley: *Safeguarding Cryptographic Keys*, Proceedings of AFIPS 1979 National Computer Conference, vol. 48, N.Y., 1979, pp. 313–317.
16. M. Blum: *Three Applications of the Oblivious Transfer*, Technical report, Dept. EECS, University of California, Berkeley, CA, 1981.
17. G. Brassard, C. Crépeau and M. Sántha: *Oblivious Transfers and Intersecting Codes*, IEEE Transaction on Information Theory , special issue on coding and complexity, Volume 42, Number 6, pp. 1769-1780, November 1996.
18. E. F. Brickell: *Some Ideal Secret Sharing Schemes*, J. Combin. Maths. & Combin. Comp. 9 (1989), pp. 105–113.
19. R. Canetti: *Studies in Secure Multiparty Computation and Applications*, Ph. D. thesis, Weizmann Institute of Science, 1995.
20. R. Canetti, U. Feige, O. Goldreich, M. Naor: *Adaptively Secure Multi-Party Computation*, Proc. ACM STOC '96, pp. 639–648.
21. R. Canetti: *Security and Composition of Multiparty Cryptographic Protocols*, draft, presented at the 1998 Weizmann Workshop on Cryptography, Weizmann Institute of Science, Rehovot, Israel, June 1998.
22. D. Chaum: *Achieving Electronic Privacy*, Scientific American, August 1992.
23. D. Chaum, I. Damgård and J. vd Graaf: *Multi-Party Computations Ensuring Secrecy of Each Party's Input and Correctness of the Output*, Proceedings of Crypto'87 volume 293 of Lecture Notes in Computer Science, pages 87-119, 16-20, Springer-Verlag, 1988.
24. D. Chaum, C. Crépeau, I. Damgård: *Multi-Party Unconditionally Secure Protocols*, Proc. of ACM STOC '88, pp. 11–19.
25. D. Chaum: *Transaction Systems to make Big Brother Obsolete*, Communications of the ACM, vol. 28, no. 10, October 1985, pp. 1030–1044.
26. D. Chaum: *Untraceable Electroic Mail, Return Addresses, and Digital Pseudonyms*, Communications of the ACM, vol. 24, no. 2, 1985, pp. 84–88.
27. B. Chor, S. Goldwasser, S. Micali, B. Awerbuch: *Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults*, Proc. IEEE FOCS '85, pp. 383–395.
28. R. Cramer: *Introduction to Secure Computation*, Lectures on Data Security– Modern Cryptology in Theory and Practice, Ivan Damgaard (Ed.), Springer LNCS, vol. 1561, Spring 1999, pp. 16–62.
29. R. Cramer, I. Damgård and U. Maurer: *General and Efficient Secure Multi-Party Computation from any Linear Secret Sharing Scheme*, February 1999. To appear in the Proceedings of EUROCRYPT '00, Brugge, Belgium, May 2000, Springer Verlag LNCS. This version includes [30]. Earlier version (BRICS Report Series

RS-97-28, "Span Programs and Secure Multi-Party Computation") presented at the 1998 Weizmann Workshop on Cryptography, Weizmann Institute of Science, Rehovot, Israel, June 1998, is obsolete.
30. R. Cramer, I. Damgård and U. Maurer: *Enforcing the Multiplication Property for Monotone Span Programs, with only Constant Overhead*, January 1999.
31. R. Cramer, I. Damgård and S. Dziembowski: *On the Complexity of Verifiable Secret Sharing and Multi-Party Computation*, to appear in the Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC '00), Portland, Oregon, May 2000.
32. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt and T. Rabin: *Efficient Multi-Party Computations with Dishonest Minority*, Proceedings of Eurocrypt '99, Springer Verlag LNCS, vol. 1592, pp.311–326, May '99.
33. R. Cramer, I. Damgård: *Zero Knowledge for Finite Field Arithmetic or: Can Zero Knowledge be for Free?*, Proc. of CRYPTO'98, Springer Verlag LNCS series, vol. 1462, pp. 424–441, 1998.
34. R. Cramer, R. Gennaro and B. Schoenmakers : *A Secure and Optimally Efficient Multi-Authority Election Scheme*, Proceedings of EUROCRYPT '97, Konstanz, Germany, Springer Verlag LNCS, vol. 1233, pp. 103–118, May 1997. Journal version: Eur. Trans. Telecom, Vol. 8, No. 5, Sept./Oct. 1997.
35. R. Cramer, M. Franklin, B. Schoenmakers, M. Yung: *Secure Secret Ballot Election Schemes with Linear Work*, Proceedings of EUROCRYPT '96, Zaragoza, Spain, Springer Verlag LNCS, vol. 1070, pp. 72–83, May 1996.
36. C. Crépeau: *Equivalence between two flavours of oblivious transfers (abstract)*, Proceedings of Crypto '87 , volume 293 of Lecture Notes in Computer Science , pages 350-354. Springer-Verlag, 1988.
37. C. Crépeau: *Correct and Private Reductions among Oblivious Transfers* PhD thesis, Department of Elec. Eng. and Computer Science, Massachusetts Institute of Technology, 1990.
38. C. Crépeau, J.vd.Graaf and A. Tapp: *Committed Oblivious Transfer and Private Multiparty Computation*, proc. of Crypto 95, Springer Verlag LNCS series.
39. C. Crépeau and J. Kilian: *Achieving oblivious transfer using weakened security assumptions*, In 29th Symp. on Found. of Computer Sci. , pages 42-52. IEEE, 1988.
40. C. Crépeau and L. Salvail: *Oblivious Verification of Common String*, CWI Quarterly (Special Issue on Cryptography), 8 (2), June 1995.
41. A. De Santis, Y. Frankel, Y. Desmedt and M. Yung: *How to Share a Function Securely*, Proceedings of 26th Annual ACM STOC, pp. 522–522, 1994.
42. Y. Desmedt: *Threshold Cryptography*, European Transactions in Telecommunication, 5 (1994), 449–457.
43. S. Even, O. Goldreich and A. Lempel: *A Randomized Protocol for Signing Contracts*, Communications of the ACM, vol. 28, 1985, pp. 637–647.
44. R. Fagin, M. Naor and P. Winkler: *Comparing Common Secret Information without Leaking it*, Communications of the ACM, vol 39, May 1996, pp. 77–85.
45. S. Fehr: *Efficient Construction of Dual MSP*, Manuscript, January 1999.
46. P. Feldman: *A practical scheme for non-interactive verifiable secret sharing*, Proceedings of 28th Annual Symposium on Foundations of Computer Science, pages 427-437, Los Angeles, California, 12-14 October 1987. IEEE.
47. P. Feldman, S. Micali: *An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement*, SIAM J. Comp. Vol. 26, No. 4, pp. 873–933, August 1997.

48. M. Fischer, S. Micali and C. Rackoff: *A Secure Protocol for Oblivious Transfer (extended abstract)*, presented at Eurocrypt '84. First published in Journal of Cryptology, 9(3):191–195, Summer 1996.
49. M. Fitzi and U. Maurer: *Efficient Byzantine Agreement Secure Against General Adversaries*, Proceedings of 12th International Symposium on Distributed Computing (DISC '98).
50. Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung: *Optimal-resilience proactive public-key cryptosystems*, Proceedings of 38th Annual Symposium IEEE FOCS pages 384-393, 1997.
51. M. Franklin: *Complexity and Security of Distributed Protocols*, Ph.D. thesis, Columbia University, New York, 1992.
52. A. Gál: *Combinatorial Methods in Boolean Function Complexity*, Ph.D.-thesis, University of Chicago, 1995.
53. Z. Galil, S. Haber and M. Yung: *Cryptographic computation: Secure fault-tolerant protocols and the public-key model*, Proceedings of Crypto '87, volume 293 of Lecture Notes in Computer Science, pages 135-155, 16-20 August 1987. Springer-Verlag, 1988.
54. J.A. Garay and Y. Moses: *Fully polynomial Byzantine agreement for n ¿ 3t processors in t + 1 rounds*, SIAM Journal on Computing, 27(1):247-290, February 1998.
55. R. Gennaro: *Theory and Practice of Verifiable Secret Sharing*, Ph.D.-thesis, MIT, 1995.
56. R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin: *Robust and efficient sharing of RSA functions*, Proceedings of CRYPTO '96, volume 1109 of Lecture Notes in Computer Science, pages 157-172, 18-22, 1996.
57. R. Gennaro, M. Rabin, T. Rabin, *Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography*, Proceedings of ACM PODC'98.
58. O. Goldreich, S. Micali and A. Wigderson: *Proofs that Yield Nothing but the Validity of the Assertion, and a Methodology of Cryptographic Protocol Design*, Proceedings IEEE FOCS'86, pp. 174–187.
59. O. Goldreich, S. Micali and A. Wigderson: *How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority*, Proc. of ACM STOC '87, pp. 218–229.
60. O. Goldreich and R. Vainish: *How to Solve any Protocol Problem: An Efficiency Improvement*, Proceedings of Crypto'87, volume 293 of Lecture Notes in Computer Science, pages 73-86, 16-20 August 1987.
61. O. Goldreich: *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, ISBN 3-540-64766-X, Springer-Verlag, Algorithms and Combinatorics, Vol. 17, 1998.
62. O. Goldreich: *Secure Multi-Party Computation* (working draft), Weizman Institute of Science, Rehovot, Israel, June 1998. Avaliable through the author's homepage `http://theory.lcs.mit.edu/ oded/`.
63. S. Goldwasser, S. Micali and C. Rackoff: *The Knowledge Complexity of Interactive Proof Systems*, Proceedings of ACM STOC'85, pp. 291–304.
64. M. Hirt, U. Maurer: *Complete Characterization of Adversaries Tolerable in General Multiparty Computations*, Proc. ACM PODC'97, pp. 25–34.
65. M. Ito, A. Saito and T. Nishizeki: *Secret Sharing Scheme Realizing General Access Structures*, Proceedings IEEE Globecom '87, pp. 99–102, 1987.

66. M. Karchmer, A. Wigderson: *On Span Programs*, Proc. of Structure in Complexity, 1993.
67. J. Kilian: *Founding Cryptography on Oblivious Transfer*, ACM STOC '88, pp. 20–31.
68. J. Kilian, S. Micali and R. Ostrovsky: *Minimum resource zero-knowledge proofs (extended abstract)*, Proceedings of 30th Annual IEEE Symposium on Foundations of Computer Science, pages 474-479, November 1989, IEEE.
69. L. Lamport, R.E. Shostak and M.C. Pease: *The Byzantine generals problem*, ACM Transactions on Programming Languages and Systems, 4(3):382-401, July 1982.
70. S. Micali and P. Rogaway:*Secure Computation*, Manuscript, Preliminary version in Proceedings of Crypto 91.
71. R. Ostrovsky and M. Yung: *How to withstand mobile virus attacks*, Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing, pages 51-59, 1991
72. T. P. Pedersen: *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*, Proc. CRYPTO '91, Springer Verlag LNCS, vol. 576, pp. 129–140.
73. T. Rabin: *A Simplified Approach to Threshold and Proactive RSA*, Proceedings of Crypto '98, Springer Verlag LNCS, vol. 1462, pp. 89–104, 1998.
74. T. Rabin: *Robust Sharing of Secrets when the Dealer is Honest or Cheating*, J. ACM, 41(6):1089-1109, November 1994.
75. T. Rabin, M. Ben-Or: *Verifiable Secret Sharing and Multiparty Protocols with Honest majority*, Proc. ACM STOC '89, pp. 73–85.
76. M. Rabin: How to Exchange Secrets by Oblivious Transfer, Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.
77. R. Rivest, A. Shamir and L. Adleman: *A Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of ACM, 21 (1978), pp. 120–126.
78. T. Sander, A. Young and M. Yung: *Non-Interactive Crypto Computing for NC1*, Proceedings of the 40th IEEE Foundations of Computer Science (FOCS '99), 1999.
79. A. Shamir: *How to Share a Secret*, Communications of the ACM 22 (1979) 612–613.
80. S. Wiesner: *Conjugate Coding*, SIGACT News, vol. 15, no. 1, 1983, pp. 78-88; Manuscript written *circa* 1970, unpublished until it appeared in SIGACT News.
81. A. Yao: *Protocols for Secure Computation*, Proc. IEEE FOCS '82, pp. 160–164.

20 Appendix A: MSP's with Multiplication

We include the following theorem and its proof from Cramer & Damgaard & Maurer [29, 30], as an extension of the material presented in Section 17.3.

Let Γ be an monotone access structure on $\{1,\ldots,n\}$, and assume that it is Q^2 (i.e. for all $A, B \notin \Gamma$, we have $A \cup B$ is *properly* contained in $\{1,\ldots,n\}$), and let $\mathcal{M}$ be a monotone span program of size m, that computes Γ. Then there exists a monotone span program *with multiplication* (see Section 17.3) of *size* $O(m)$ that computes Γ as well. [21]

Monotone span programs [66] (MSP's) are the basis for linear secret sharing schemes (LSSS): each of the shares in a secret is equal to a fixed linear combination of the secret and random (field-) elements chosen by the dealer. The MSP underlying an LSSS is a linear algebraic device [66] that "computes" the access

[21] Moreover, it can be efficiently constructed from the original monotone span program.

structure of the LSSS. On the other hand, each MSP gives rise to an LSSS. Hence, one can identify an LSSS with its underlying MSP (see Section 17.3).

In [29, 32] it is shown that given just an LSSS with Q^2-access structure Γ, there exist a general multi-party computation protocol that is secure with respect to Γ and that has efficiency polynomial in the complexity of the LSSS. The theorem stated above is essential in the proof.

One can define the algebraic complexity of a monotone access structure as the complexity of the "smallest" MSP computing it. There is a tight and obvious relationship between the algebraic complexity of a monotone access structure and the "most efficient" LSSS for that access structure.

By their very nature, LSSS support secure distributed computation of *sums* of shared secrets: each player locally adds the shares he holds in the secrets involved.

However, it is far from clear how to securely compute the *product* of two shared secrets based on a given LSSS: in general, secure multiplication appears to be an inherently *non-linear* phenomenon. Of course, these elementary arithmetical operations form the basis for general secure multi-party computation of arbitrary functions (where the security is with respect to the access structure of the LSSS).

Here, security is defined with respect to the access structure of the MSP. This means that even if an adversary picks a single non-qualified set of players and corrupts them, the protocols will still be secure.

In [29], the problem of multiplying secrets is handled by requiring that the monotone span program underlying the LSSS has a special multiplication property. It is shown in [29] how to use this property to perform secure multi-party computation, even in the presence of *malicious* adversaries.

So given an access structure that is computable by an MSP with multiplication, the question arises how the algebraic complexity of the access structure in this *enhanced* model of MSP's with multiplication compares to its ordinary MSP complexity. It is certainly not smaller, but in theory it may be much larger and perhaps even with super-polynomial differences.

Note that as a special case of the general result of [64], security with respect to a Q^2-access structure is the best one can hope for anyway. Therefore, if an LSSS supports general multi-party computation, the corresponding access structure must be Q^2.

Theorem 1. (Cramer/Damgaard/Maurer *[29, 30]) Let Γ be a monotone access structure computed by an MSP of size d. Then there exists an MSP with multiplication that computes Γ and that has size $O(d)$, provided that Γ is Q^2. Moreover, the access structure computed by any MSP with multiplication is Q^2.*

Proof: The last claim in the theorem is easy to prove. Let $\mathcal{M} = (K, M, \psi)$ be an MSP with multiplication. Suppose its access structure Γ (on $\{1, \ldots, n\}$) is *not* Q^2.

Then there exist sets $A, B \notin \Gamma$ such that $A \cup B = \{1, \ldots, n\}$. It is easy to see (by a simple duality argument from linear algebra, see Section 16.1) that

$A \notin \Gamma$ iff there exists $\boldsymbol{\kappa} \in K^e$ such that $M_A\boldsymbol{\kappa} = \mathbf{0}$ and its first coordinate κ_1 is equal to 1.

Let $\mathbf{s}$ be any full set of shares in the LSSS corresponding to the MSP, with secret s. Since it has multiplication we have $\langle \mathbf{r}, \mathbf{s} * M\boldsymbol{\kappa} \rangle = s \cdot 1 = s$. But $\mathbf{s} * M\boldsymbol{\kappa}$ is zero in coordinates corresponding to A. Therefore, the players in B can compute $\mathbf{s} * \boldsymbol{\kappa}$ and hence s on their own (since $\boldsymbol{\kappa}$ depends only on A). But this implies $B \in \Gamma$ which is a contradiction.

Before proving the first claim in the theorem, we make some observations. Let Γ_0 and Γ_1 be any access structures, computed by arbitrary respective MSP's $\mathcal{M}_0 = (K, M_0, \psi)$ and $\mathcal{M}_1 = (K, M_1, \psi)$, where M_0 and M_1 are $d \times e_0$ and $d \times e_1$ matrices, respectively, where the mapping ψ is identical for both MSPs, and where the target vector is $(1, 0, \ldots, 0)$ of length e_0 and e_1, respectively. Let $\Gamma = \Gamma_0 \cup \Gamma_1$ be the union of the two access structures. Now suppose that the matrices M_0 and M_1 satisfy

$$M_0^T M_1 = E, \tag{1}$$

where E is $e_o \times e_1$ matrix that is zero everywhere, except in its upper-left corner where the entry is 1. Below we show that (1) implies that one can construct an MSP *with multiplication* (actually a generalization thereof) whose *size* is of the same order as that of $\mathcal{M}_0$ and $\mathcal{M}_1$, and which computes Γ.

Consider the following straightforward LSSS for Γ. The dealer shares the secret $s \in K$ using LSSS$_0$ *and* LSSS$_1$, given by $\mathcal{M}_0$ and $\mathcal{M}_1$, respectively. That is, he selects a pair of vectors $(\mathbf{b}_0, \mathbf{b}_1)$ at random, except that the first entries are both s: $\langle \boldsymbol{t}, \mathbf{b}_0 \rangle = \langle \boldsymbol{t}, \mathbf{b}_1 \rangle = s$. Then he computes the pair of vectors $(\mathbf{s}_0, \mathbf{s}_1) = (M_0\mathbf{b}_0, M_1\mathbf{b}_1)$, and sends for $i = 1, \ldots, n$ the i-th coordinates of $\mathbf{s}_0$ and $\mathbf{s}_1$ to player $P_{\psi(i)}$. It is clear that a subset A of the players can reconstruct s from their joint shares if and only if A is qualified with respect to either Γ_0 or Γ_1, i.e. with respect to Γ.

How can we multiply two shared secrets? Assume that $s' \in K$ is a secret with full set of shares $(\mathbf{s}_0', \mathbf{s}_1') = (M_0\mathbf{b}_0', M_1\mathbf{b}_1')$, where $\langle \boldsymbol{t}, \mathbf{b}_0' \rangle = \langle \boldsymbol{t}, \mathbf{b}_1' \rangle = s'$. From (1) we have

$$\langle \mathbf{1}, \mathbf{s}_0 * \mathbf{s}_1' \rangle = \mathbf{s}_0^T \mathbf{s}_1' = \mathbf{b}_0^T M_0^T M_1 \mathbf{b}_1' = \mathbf{b}_0^T E \mathbf{b}_1' = ss', \tag{2}$$

where $\mathbf{1}$ denotes the all-one vector of appropriate length. Note that for each i, the shares in the i-th coordinate of $\mathbf{s}_0$ and the i-th coordinate of $\mathbf{s}_1'$ are held by the same player.

In brief, there is a fixed linear combination (here consisting of all 1's) over products of shares in s and s' locally computed by each player, which results in the product of the secrets. This is sufficient for securely multiplying s and s', since each player can re-share these locally computed values, and add the newly received shares to obtain his share in ss'. This technical idea of securely multiplying two shared secrets was first presented in the case of threshold secret sharing (Shamir's scheme) by Gennaro & Rabin & Rabin [57].

The above description of how two shared values s and s' can be multiplied does not match exactly our definition of the multiplication property of an

LSSS because the players actually multiply shares of s in LSSS_0 with shares of s' in LSSS_1. However, observe that by an argument symmetric to that leading to (2), we also have $\langle \mathbf{1}, \mathbf{s}'_0 * \mathbf{s}_1 \rangle = ss'$, and this symmetry can be exploited to define an MSP (or LSSS) for Γ with the multiplication property as defined in Section 17.3.

We are now ready to prove the first claim in the theorem. Recall the definition of the dual access structure Γ^* of Γ: $A \in \Gamma^*$ if and only if $\overline{A} \notin \Gamma$, where $\overline{A}$ denotes $\{1, \ldots, n\} \setminus A$. Let Γ be a monotone access structure that is Q^2 (this is equivalent to saying that for all $A \notin \Gamma$ we have $\overline{A} \in \Gamma$). It follows immediately from the definition that Γ is Q^2 if and only if $\Gamma^* \subseteq \Gamma$.

Let $\mathcal{M} = (K, M, \psi)$ be an MSP computing the access structure Γ, with target vector $\boldsymbol{t} = (1, 0, \ldots, 0)$. We set $\Gamma_0 = \Gamma$, and $\mathcal{M}_0 = \mathcal{M}$. It is now sufficient to present $\mathcal{M}_1$ with access structure $\Gamma_1 = \Gamma^*$ (the dual) so that the pair M_0, M_1 satisfies equation 1.

In [52] a construction is presented which, given an MSP $\mathcal{N} = (K, N, \psi)$ of size d with access structure Γ (and target vector $(1, \ldots, 1)$), yields a "dual" MSP $\mathcal{N}^* = (K, N^*, \psi)$ with access structure Γ^* (and target vector all-one). In particular, N^* has the same number of rows and labeling as N.

The construction is as follows. N^* has also d rows and the same labeling as N and consists of one column for each minimal $A \in \Gamma$, namely any reconstruction vector $\boldsymbol{\lambda}$ for for the set A for the LSSS (satisfying $\boldsymbol{\lambda}^T N = (1, \ldots, 1)$ and $\boldsymbol{\lambda}_{\overline{A}} = \mathbf{0}$. The matrix N^* has generally exponentially many columns, but it is easy to see that any linearly independent generating subset of them (at most d) will do.[22] It follows from the construction that $N^T N^*$ is an all-one matrix, which we call U.

In our case the target vector of $\mathcal{M}$ is $\boldsymbol{t} = (1, 0, \ldots, 0)$ instead of $(1, \ldots, 1)$, but we noted earlier that the target vector can be transformed. This is achieved by adding the first column of M to every other column of M. More formally, let H be the isomorphism that sends an e-(column) vector to an e-(column) vector by adding its first coordinate to each other coordinate. Write $N = MH^T$. Then the MSP $\mathcal{N} = (K, N, \psi)$ is as $\mathcal{M}$ except that the target vector is all-one. Now let $\mathcal{N}^* = (K, N^*, \psi)$ be its dual MSP according to [52]. Finally write $M^* = N^*(H^{-1})^T$. Then $\mathcal{M}^* = (K, M^*, \psi)$ has target vector $\boldsymbol{t}$ and computes Γ^*. Observe that $M^T M^* = H^{-1} U (H^{-1})^T = E$, as desired.

□

[22] This construction process if used directly is not efficient, but the matrix N^* can be constructed efficiently, without enumerating all columns of the construction [45].