Robust Combiners

Krzysztof Pietrzak (CWI Amsterdam)

Chennai, December 13th, 2007


U Locks
Cable Locks
Cables
Chain Locks
Chains
Cuff Locks
Python Locks
The Bike Club

Twist a Pen, Open a Lock

Leander Kahney 09.17.04 | 2:00 AM

After cutting four small slits in the end of the pen's barrel to ease it in, the lock opened with a single twist. View Slideshow

A 50-year-old lock design was rendered useless last week when a brief post to an internet forum revealed the lock can be popped open with a cheap plastic pen.

On Sunday, bike enthusiast and network security consultant Chris Brennan described opening an expensive Kryptonite bike lock using a ballpoint pen.

"Your brand new U-Lock is not safe," warned Brennan in a note posted to Bike Forums.

Wired News tested Brennan's claims. A brand new Kryptonite Evolution 2000 was opened in seconds using a Bic pen. After cutting four small slits in the end of the pen's barrel to ease it in, the lock opened with a single twist.

Brennan, 25, of San Francisco, said he successfully opened two Kryptonite locks, an Evolution 2000 and an older Kryptonite Mini lock.

Subsequent posts to Bike Forums and other websites report the vulnerability applies to many of the company's cylindrical-lock products, including some from Kryptonite's vaunted New York series.

Use two different locks, with separate locking mechanisms. Thieves carry tools that will either snip cables, or pry-apart U-locks but rarely both. A cable-lock and a U-lock together are very secure.

Robust Combiner: Informal Definition

Definition (Robust $(1,2)$-Combiner for XXX)

A combiner for XXX is a construction, which given two candidate implementations of XXX, is a secure realization of XXX if at least one of the two candidates is secure.

Robust Combiner: Informal Definition

Definition (Robust $(1, 2)$-Combiner for XXX)

A combiner for XXX is a construction, which given two candidate implementations of XXX, is a secure realization of XXX if at least one of the two candidates is secure.

Definition (Robust (k, ℓ)-Combiner for XXX)

A combiner for XXX is a construction, which given ℓ candidate implementations of XXX, is a secure realization of XXX if at least k one of the two candidates is secure.

Related Concept is *Amplification*: Combine many instantiations of the same candidate, if a single instantiation is insecure with probability ϵ, then k instantiations will be insecure with probability $\ll \epsilon$, ideally $O(\epsilon^k)$.

Outline

Part 1: Robust Combiners for Cryptographic Primitives: Definitions and Constructions

Part 2: $(1, n)$-Combiners from $(1, 2)$-Combiners and Universal Schemes

Part 3: Combiners for Collision Resistance

Robust Combiners for Cryptographic Primitives: Definitions and Constructions

A. Herzberg, On cryptographic tolerance, CT-RSA 2005

D. Harnik, J.Kilian, M.Naor, O.Reingold, A.Rosen, On Robust Combiners for Oblivious Transfer and other Primitives, EUROCRYPT 2005

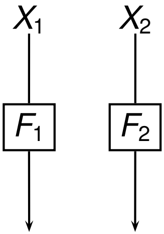
A Combiner For One-Way Functions

$F : \mathcal{X}_n \to \mathcal{Y}_n$ is a One Way Function if for all efficient A

$$\Pr_{X \leftarrow \mathcal{X}_n}[A(F(X)) \to X' \textit{ where } F(X') = F(X)] = \mathrm{negl}(n)$$

A Combiner For One-Way Functions

$$C^{F_1, F_2}(X_1, X_2) = F_1(X_1) \| F_2(X_2)$$



A Combiner For One-Way Functions

Claim

$C^{F_1,F_2}(X_1, X_2) = F_1(X_1) \| F_2(X_2)$ is a robust combiner.

Proof: Let A be an adversary who breaks C^{F_1,F_2}, i.e. for some non-negligeable $\delta(.)$

$$\Pr_{X_1,X_2 \leftarrow \mathcal{X}_n}[A(F_1(X_1) \| F_2(X_2)) \to F_1^{-1}(X_1) \| F_2^{-1}(X_2)] = \delta(n)$$

We can invert F_1 and F_2 with one call to A with prob. $\delta(.)$.
On input $Y = F_1(X)$:

- sample $X' \leftarrow \mathcal{X}_n$, set $Y' := F_2(X')$.
- Invoke $A(Y \| Y') \to Z \| Z'$
- Output Z

Note that $Pr[F_1(Z) = Y] = \delta(n)$. $\square$

Formal Definition

Definition (Cryptographic Primitive)

A **primitive** $\mathcal{P}$ is a triplet $\langle F_{\mathcal{P}}, \mathcal{A}_{\mathcal{P}}, R_{\mathcal{P}} \rangle$, where $F_{\mathcal{P}}$ is a set of functions $f : \{0,1\}* \rightarrow \{0,1\}^*$ defining the functionality of $\mathcal{P}$, $\mathcal{A}_{\mathcal{P}}$ is the class of adversary machines and $R_{\mathcal{P}}$ is a relation over pairs $\langle f, A \rangle$, including machines $A \in \mathcal{A}_{\mathcal{P}}$ that **break** functions $f \in F_{\mathcal{P}}$. We say that f **implements** $\mathcal{P}$ if $f \in F_{\mathcal{P}}$ and is computable by a PPTM. A **secure implementation** is an f that no $A \in \mathcal{A}_{\mathcal{P}}$ breaks. The primitive $\mathcal{P}$ **exists** if there exists an implementation of $\mathcal{P}$ that is secure.

Example: OWF

The primitive "one-way-function" $\langle F_{OWF}, \mathcal{A}_{OWF}, R_{OWF} \rangle$.

- F_{OWF} are all functions $\{0,1\}^* \to \{0,1\}^*$.
- $\mathcal{A}_{OWF}$ are all functions $\{0,1\}^* \to \{0,1\}^*$ computable by a PPTM.
- $f \in F_{OWF}$ **implements** a OWF if it is computable by a PPTM.
- If f implements a OWF and $A \in \mathcal{A}_{OWF}$ then $\langle f, A \rangle \in R_{OWF}$ (i.e. A **breaks** f) if

$$Pr_{x \in \{0,1\}^n}[A(f(x)) = f^{-1}(x)] \neq \text{negl}(n)$$

Thus $f \in F_{OWF}$ is a **secure** implementation of a OWF if for all $A \in \mathcal{A}_{OWF}$

$$Pr_{x \in \{0,1\}^n}[A(f(x)) = f^{-1}(x)] = \text{negl}(n)$$

On Implementability

In general, it is undecidable if a candidate scheme f implements $\mathcal{P}$ (i.e. whether it is computable by a PPTM).

On Implementability

In general, it is undecidable if a candidate scheme f implements $\mathcal{P}$ (i.e. whether it is computable by a PPTM).

This is not the problem in practice: checking whether any "reasonable" candidate scheme f implements some primitive $\mathcal{P}$ can usually be done unconditionally.

The concern is solely if f **securely** *implements $\mathcal{P}$.*

Formal Definitions: Robust Combiners

Definition ((k, n)-Robust Combiner)

Fix representation for cryptographic primitive $\mathcal{P}$. A (k, n)-robust combiner for $\mathcal{P}$ is a PPTM that gets n candidate schemes as inputs and implements $\mathcal{P}$ s.t.

- If at least k of the candidates securely implement $\mathcal{P}$, so does the combiner.
- The running time of the combiner is polynomial in a security parameter in n.

Robust (k, n) combiner for $\mathcal{P}$ exists if $\mathcal{P}$ exists, as combiner may ignore the inputs and simply implement $\mathcal{P}$ securely.

Formal Definitions: Robust Combiners

Definition (Black-Box (k, n)-Robust Combiner)

C is a black-box (k, n)-Robust Combiner for $\mathcal{P}$ if it is a (k, n)-Robust Combiner where

- The implementations is black-box: C get access to the candidates via oracle calls.
- The proof is black-box: for all candidates there exists an oracle PPTM R s.t. if A breaks the combiner R^A breaks the candidate.

All known costruction of combinres are black-box. Non-black box constructions are very rare in crypto in general, the few known examples are extremely inefficient. In some cases one can *rule out* the existence of black-box combiners.

Combiner for OWFs gives a Combiner for all Primitives equivalent to OWFs.

- Pseudorandom Generators/Functions/Permutations
- Bit Commitments
- Message Authentication Codes
- Digital Signatures

Combiner for any Primitive Equivalent to OWFs

Let $\mathcal{P} \in \{\textit{PRG}, \textit{PRF}, \textit{PRP}, \textit{BC}, \textit{MAC}\}$ or any other primitive equivalent to OWFs.

Combiner for any Primitive Equivalent to OWFs

Let $\mathcal{P} \in \{PRG, PRF, PRP, BC, MAC\}$ or any other primitive equivalent to OWFs.

- Given to candidates P_1, P_2 for $\mathcal{P}$, construct F_1, F_2 such that F_i is a secure OWF is P_i is a secure $\mathcal{P}$.

Combiner for any Primitive Equivalent to OWFs

Let $\mathcal{P} \in \{PRG, PRF, PRP, BC, MAC\}$ or any other primitive equivalent to OWFs.

- Given to candidates P_1, P_2 for $\mathcal{P}$, construct F_1, F_2 such that F_i is a secure OWF is P_i is a secure $\mathcal{P}$.
- Let $F(.)$ be the combined OWF $F_1(.)\|F_2(.)$.

Combiner for any Primitive Equivalent to OWFs

Let $\mathcal{P} \in \{PRG, PRF, PRP, BC, MAC\}$ or any other primitive equivalent to OWFs.

- Given to candidates P_1, P_2 for $\mathcal{P}$, construct F_1, F_2 such that F_i is a secure OWF is P_i is a secure $\mathcal{P}$.
- Let $F(.)$ be the combined OWF $F_1(.)\|F_2(.)$.
- Construct $\mathcal{P}$ from F.

Combiner for any Primitive Equivalent to OWFs

Let $\mathcal{P} \in \{PRG, PRF, PRP, BC, MAC\}$ or any other primitive equivalent to OWFs.

- Given to candidates P_1, P_2 for $\mathcal{P}$, construct F_1, F_2 such that F_i is a secure OWF is P_i is a secure $\mathcal{P}$.
- Let $F(.)$ be the combined OWF $F_1(.)\|F_2(.)$.
- Construct $\mathcal{P}$ from F.

Just of theoretical interest, as reduction from OWF to other primitives are extremely inefficient.

Combiner for any Primitive Equivalent to OWFs

Let $\mathcal{P} \in \{PRG, PRF, PRP, BC, MAC\}$ or any other primitive equivalent to OWFs.

- Given to candidates P_1, P_2 for $\mathcal{P}$, construct F_1, F_2 such that F_i is a secure OWF is P_i is a secure $\mathcal{P}$.
- Let $F(.)$ be the combined OWF $F_1(.) \| F_2(.)$.
- Construct $\mathcal{P}$ from F.

Just of theoretical interest, as reduction from OWF to other primitives are extremely inefficient.
Fortunately, for all the above primitives, there are efficient $(1, 2)$ combiners...

Combiner for any Primitive Equivalent to OWFs

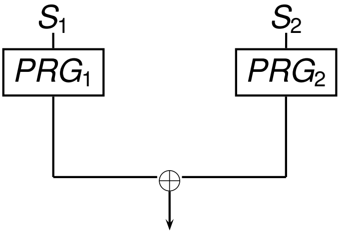
Let $\mathcal{P} \in \{PRG, PRF, PRP, BC, MAC\}$ or any other primitive equivalent to OWFs.

- Given to candidates P_1, P_2 for $\mathcal{P}$, construct F_1, F_2 such that F_i is a secure OWF is P_i is a secure $\mathcal{P}$.
- Let $F(.)$ be the combined OWF $F_1(.)\|F_2(.)$.
- Construct $\mathcal{P}$ from F.

Just of theoretical interest, as reduction from OWF to other primitives are extremely inefficient.
Fortunately, for all the above primitives, there are efficient $(1, 2)$ combiners... except for Bit-Commitment.

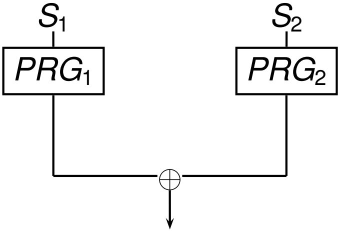
Combiner for Pseudorandom Generators

$$C^{PRG_1,PRG_2}(S_1, S_2) = PRG_1(S_1) \oplus PRG_2(S_2)$$



Combiner for Pseudorandom Generators

$$C^{PRG_1,PRG_2}(S_1, S_2) = PRG_1(S_1) \oplus PRG_2(S_2)$$

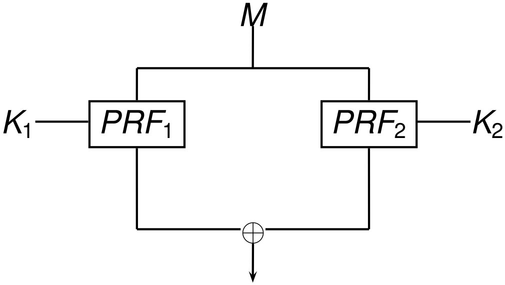


This combiner plays a crucial role in the classical construction of a PRG from OWF.

J.Håstad, R.Impagliazzo, L.A.Levin, M.Luby: A Pseudorandom Generator from any One-way Function. SIAM J. Comput. 1999

Combiner for Pseudorandom Functions

$$C^{PRF_1,PRF_2}([K_1, K_2], M) = PRF_1(K_1, M) \oplus PRF_2(K_2, M)$$



Combiners for Pseudorandom Permutations

$$C^{PRP_1,PRP_2}([K_1,K_2],M) = PRP_2(K_2, PRP_1(K_1,M))$$

$$M \longrightarrow \underset{\substack{| \\ K_1}}{\boxed{PRP_1}} \longrightarrow \underset{\substack{| \\ K_2}}{\boxed{PRP_2}} \longrightarrow$$

Combiner for Message Authentication Codes

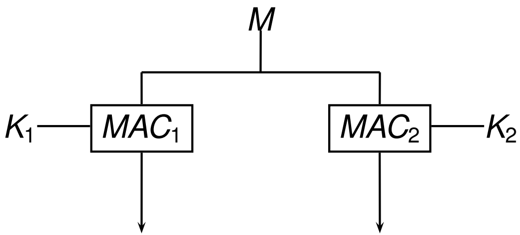
$F : \mathcal{K}_n \times \mathcal{X}_n \to \mathcal{Y}_n$ is a secure MAC if for all efficient A

$$\Pr_{K \leftarrow \mathcal{K}_n}[A^{F(K,.)} \to (\phi, M) \wedge \phi = F(K, M)] = \mathrm{negl}(n)$$

Here A is not allowed to query $F(K, .)$ on its output M.

Combiner for Message Authentication Codes

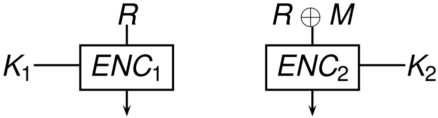
$$C^{MAC_1, MAC_2}([K_1, K_2], M) = MAC_1(K_1, M) \| MAC_2(K_2, M)$$



Combiner for Encryption

$$C^{ENC_1, ENC_2}([K_1, K_2], M) = R \| ENC_1(K_1, R) \| ENC_2(K_2, M \oplus R)$$

Where a fresh random R is picked for every encryption.



ASMUTH, C. A., AND BLAKLEY, G.R. An efficient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems. Comput. Math. Appl. 7 1981.

Collision Resistant Hash Functions

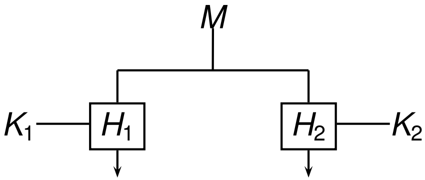
$H : \mathcal{K}_n \times \mathcal{X}_n \to \mathcal{Y}_n$ is a CRHF if for all efficient A

$$\Pr_{K \leftarrow \mathcal{K}}[A(K) = M, M' \textit{ where } H_K(M) = H_K(M')] = \mathsf{negl}(n)$$

CRHFs are not known to be equivalent to OWFs, in the sense that there exists no black-box construction of CRHFs from OWFs (Simon EC'98).

Combiner For CRHFs

$$C^{H_1,H_2}([K_1,K_2],M) = H_1(K_1,M) \| H_2(K_2,M)$$



Note that any collision M, M' for $C^{H_1,H_2}([K_1,K_2],.)$ is also a collision for $H_1(K_1,.)$ and $H_2(K_2,.)$.

Bit-Commitment

A Bit-Commitment Scheme is a function

$$BC : \{0, 1\} \times \mathcal{R}_n \to \mathcal{C}_n$$

Binding: It is hard to find r, r' where

$$BC(0, r) = BC(1, r')$$

Hiding: For uniformly random r, $BC(0, r)$ and $BC(1, r)$ are indistinguishable.

Bit-Commitment

A Bit-Commitment Scheme is a function

$$BC : \{0,1\} \times \mathcal{R}_n \to \mathcal{C}_n$$

Binding: It is hard to find r, r' where

$$BC(0,r) = BC(1,r')$$

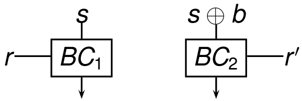
Hiding: For uniformly random r, $BC(0,r)$ and $BC(1,r)$ are indistinguishable.

Perfectly Binding: $\neg\exists r, r' : BC(0,r) = BC(1,r')$

Perfectly Hiding: $\Delta(BC(0,r), BC(1,r)) = 0$

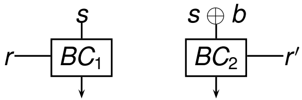
A BC scheme can be either perfectly binding or perfectly hiding, but not both.

Combiner for the Hiding Property



$$C_H^{BC_1,BC_2}(b, [r, r', s]) = BC_1(s, r) \| BC_2(b \oplus s, r')$$

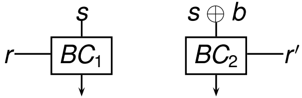
Combiner for the Hiding Property



$$C_H^{BC_1,BC_2}(b,[r,r',s]) = BC_1(s,r) \| BC_2(b \oplus s, r')$$

Not Binding: BC_1 or BC_2 not binding $\Rightarrow$ $C_H^{BC_1,BC_2}$ not binding.

Combiner for the Hiding Property

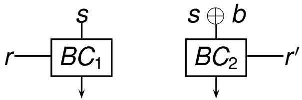


$$C_H^{BC_1,BC_2}(b,[r,r',s]) = BC_1(s,r) \| BC_2(b \oplus s, r')$$

Not Binding: BC_1 or BC_2 not binding $\Rightarrow C_H^{BC_1,BC_2}$ not binding.

Hiding: $C_H^{BC_1,BC_2}$ is hiding if either BC_1 or BC_2 is hiding.

Combiner for the Hiding Property



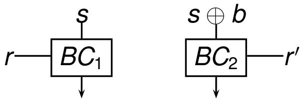
$$C_H^{BC_1,BC_2}(b, [r, r', s]) = BC_1(s, r) \| BC_2(b \oplus s, r')$$

Not Binding: BC_1 or BC_2 not binding $\Rightarrow$ $C_H^{BC_1,BC_2}$ not binding.
Hiding: $C_H^{BC_1,BC_2}$ is hiding if either BC_1 or BC_2 is hiding.

- Assume $C_H^{BC_1,BC_2}$ is not hiding: $\exists$ efficient A

$$Pr[A(BC_1(s, r) \| BC_2(b \oplus s, r')) = b] = 1/2 + \delta$$

Combiner for the Hiding Property



$$C_H^{BC_1,BC_2}(b,[r,r',s]) = BC_1(s,r)\|BC_2(b \oplus s, r')$$

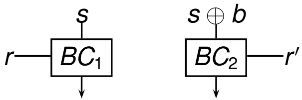
Not Binding: BC_1 or BC_2 not binding $\Rightarrow$ $C_H^{BC_1,BC_2}$ not binding.
Hiding: $C_H^{BC_1,BC_2}$ is hiding if either BC_1 or BC_2 is hiding.

- Assume $C_H^{BC_1,BC_2}$ is not hiding: $\exists$ efficient A

$$Pr[A(BC_1(s,r)\|BC_2(b \oplus s, r')) = b] = 1/2 + \delta$$

- To break BC_1: Given $com = BC_1(b,r)$, sample s, r'.

Combiner for the Hiding Property



$$C_H^{BC_1,BC_2}(b,[r,r',s]) = BC_1(s,r) \| BC_2(b \oplus s, r')$$

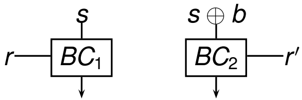
Not Binding: BC_1 or BC_2 not binding $\Rightarrow C_H^{BC_1,BC_2}$ not binding.
Hiding: $C_H^{BC_1,BC_2}$ is hiding if either BC_1 or BC_2 is hiding.

- Assume $C_H^{BC_1,BC_2}$ is not hiding: $\exists$ efficient A

$$Pr[A(BC_1(s,r) \| BC_2(b \oplus s, r')) = b] = 1/2 + \delta$$

- To break BC_1: Given $com = BC_1(b,r)$, sample s, r'.
- Call $A(com \| BC_2(s,r')) \to d$ and output $b' = d \oplus s$.

Combiner for the Hiding Property



$$C_H^{BC_1,BC_2}(b,[r,r',s]) = BC_1(s,r)\|BC_2(b\oplus s,r')$$

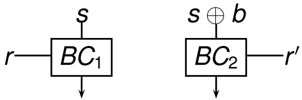
Not Binding: BC_1 or BC_2 not binding $\Rightarrow$ $C_H^{BC_1,BC_2}$ not binding.
Hiding: $C_H^{BC_1,BC_2}$ is hiding if either BC_1 or BC_2 is hiding.

- Assume $C_H^{BC_1,BC_2}$ is not hiding: $\exists$ efficient A

$$Pr[A(BC_1(s,r)\|BC_2(b\oplus s,r')) = b] = 1/2+\delta$$

- To break BC_1: Given $com = BC_1(b,r)$, sample s, r'.
- Call $A(com\|BC_2(s,r')) \to d$ and output $b' = d \oplus s$.
- $\Pr[b = b'] = 1/2+\delta$. BC_2 is broken similarly.

Combiner for the Hiding Property

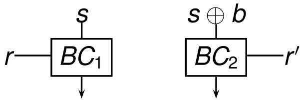


$$C_H^{BC_1,BC_2}(b,[r,r',s]) = BC_1(s,r) \| BC_2(b \oplus s, r')$$

Not Binding: BC_1 or BC_2 not binding $\Rightarrow C_H^{BC_1,BC_2}$ not binding.

Hiding: $C_H^{BC_1,BC_2}$ is hiding if either BC_1 or BC_2 is hiding.

Combiner for the Hiding Property



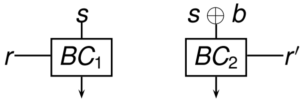
$$C_H^{BC_1,BC_2}(b,[r,r',s]) = BC_1(s,r)\|BC_2(b\oplus s,r')$$

Not Binding: BC_1 or BC_2 not binding $\Rightarrow$ $C_H^{BC_1,BC_2}$ not binding.

Hiding: $C_H^{BC_1,BC_2}$ is hiding if either BC_1 or BC_2 is hiding.

Preserving for Binding: If BC_1 and BC_2 are binding, so is $C_H^{BC_1,BC_2}$.

Combiner for the Hiding Property

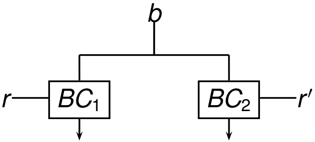


$$C_H^{BC_1,BC_2}(b,[r,r',s]) = BC_1(s,r)\|BC_2(b\oplus s,r')$$

Not Binding: BC_1 or BC_2 not binding $\Rightarrow C_H^{BC_1,BC_2}$ not binding.
Hiding: $C_H^{BC_1,BC_2}$ is hiding if either BC_1 or BC_2 is hiding.
Preserving for Binding: If BC_1 and BC_2 are binding, so is $C_H^{BC_1,BC_2}$.

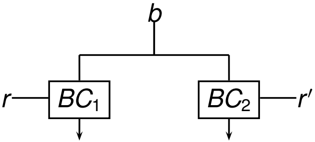
Combiner C_H for the hiding property, is a robust combiner for perfectly binding BC (as here binding is unconditional).

Combiner for the Binding Property



$$C_B^{BC_1,BC_2}(b,[r,r']) = BC_1(b,r) \| BC_2(b,r')$$

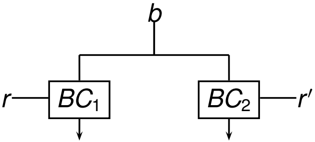
Combiner for the Binding Property



$$C_B^{BC_1,BC_2}(b,[r,r']) = BC_1(b,r)\|BC_2(b,r')$$

Not Hiding: BC_1 or BC_2 not hiding $\Rightarrow C_B^{BC_1,BC_2}$ not hiding.

Combiner for the Binding Property

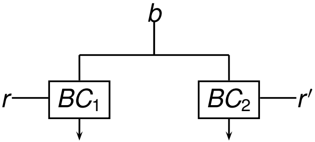


$$C_B^{BC_1,BC_2}(b,[r,r']) = BC_1(b,r) \| BC_2(b,r')$$

Not Hiding: BC_1 or BC_2 not hiding $\Rightarrow C_B^{BC_1,BC_2}$ not hiding.

Binding: $C_B^{BC_1,BC_2}$ is binding if either BC_1 or BC_2 is binding.

Combiner for the Binding Property



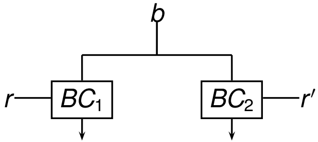
$$C_B^{BC_1,BC_2}(b,[r,r']) = BC_1(b,r)\|BC_2(b,r')$$

Not Hiding: BC_1 or BC_2 not hiding $\Rightarrow$ $C_B^{BC_1,BC_2}$ not hiding.
Binding: $C_B^{BC_1,BC_2}$ is binding if either BC_1 or BC_2 is binding.

- Assume $C_B^{BC_1,BC_2}$ is not binding: $\exists$ efficient A

  $Pr[A \rightarrow (r,r',s,s') : BC_1(0,r)\|BC_2(0,r') = BC_1(1,s)\|BC_2(1,s')$

Combiner for the Binding Property



$$C_B^{BC_1,BC_2}(b,[r,r']) = BC_1(b,r) \| BC_2(b,r')$$

Not Hiding: BC_1 or BC_2 not hiding $\Rightarrow$ $C_B^{BC_1,BC_2}$ not hiding.

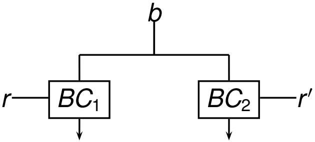
Binding: $C_B^{BC_1,BC_2}$ is binding if either BC_1 or BC_2 is binding.

- Assume $C_B^{BC_1,BC_2}$ is not binding: $\exists$ efficient A

  $$Pr[A \to (r,r',s,s') : BC_1(0,r) \| BC_2(0,r') = BC_1(1,s) \| BC_2(1,s')$$

- This A breaks BC_1 as $BC_1(0,r) = BC_1(1,s)$, and it breaks BC_2 as $BC_2(0,r') = BC_2(1,s')$.

Combiner for the Binding Property

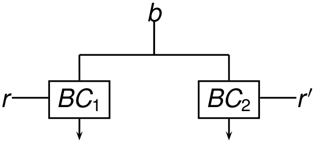


$$C_B^{BC_1,BC_2}(b,[r,r']) = BC_1(b,r) \| BC_2(b,r')$$

Not Hiding: BC_1 or BC_2 not hiding $\Rightarrow C_B^{BC_1,BC_2}$ not hiding.

Binding: $C_B^{BC_1,BC_2}$ is binding if either BC_1 or BC_2 is binding.

Combiner for the Binding Property



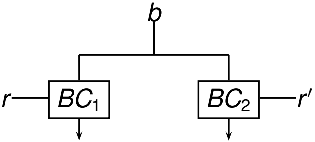
$$C_B^{BC_1, BC_2}(b, [r, r']) = BC_1(b, r) \| BC_2(b, r')$$

Not Hiding: BC_1 or BC_2 not hiding $\Rightarrow$ $C_B^{BC_1, BC_2}$ not hiding.

Binding: $C_B^{BC_1, BC_2}$ is binding if either BC_1 or BC_2 is binding.

Preserving for Hiding: If BC_1 and BC_2 are hiding, so is $C_B^{BC_1, BC_2}$.

Combiner for the Binding Property



$$C_B^{BC_1,BC_2}(b,[r,r']) = BC_1(b,r)\|BC_2(b,r')$$

Not Hiding: BC_1 or BC_2 not hiding $\Rightarrow C_B^{BC_1,BC_2}$ not hiding.
Binding: $C_B^{BC_1,BC_2}$ is binding if either BC_1 or BC_2 is binding.
Preserving for Hiding: If BC_1 and BC_2 are hiding, so is $C_B^{BC_1,BC_2}$.

Combiner C_B for the binding property, is a robust combiner for perfectly hiding BC (as here hiding is unconditional).

Open Problem

Efficient Robust (1,2)-Combiner for general BC (inefficient exist via OWFs).

Open Problem

Efficient Robust (1,2)-Combiner for general BC (inefficient exist via OWFs).

For any $t \in \mathbb{N}$, Efficient Robust (t+1,2t+1)-Combiner Exist (Herzberg). We will prove the case $t = 1$.

Robust (2,3)-Combiner for BC

Given: BC_1, BC_2, BC_3 two of which are secure (binding & hiding).

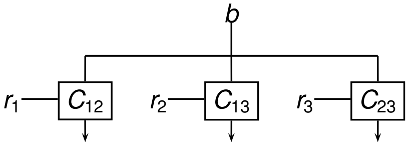
Let $C_{12} = C_H^{BC_1,BC_2}, C_{13} = C_H^{BC_1,BC_3}, C_{23} = C_H^{BC_2,BC_3}$, where

$$C_H^{BC_1,BC_2}(b, [r, r', s]) = BC_1(s, r) \| BC_2(b \oplus s, r')$$

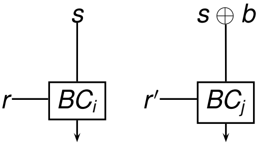
is the combiner for the hiding property.

The following is a robust $(2, 3)$-combiner for BC.

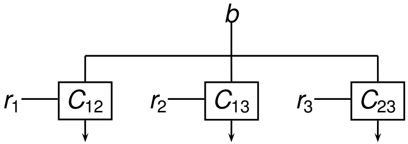
$$C^{BC_1,BC_2,BC_3}(b, [r, r', r'']) = C_{12}(b, r) \| C_{13}(b, r') \| C_{23}(b, r'')$$



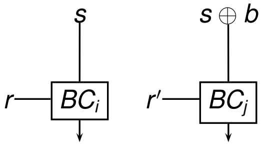
where C_{ij} is



$$C^{BC_1,BC_2,BC_3}(b,r) = C_B^{C_{12},C_{13},C_{23}}(b,r) \quad \text{where} \quad C_{ij} = C_H^{BC_i,BC_j}$$



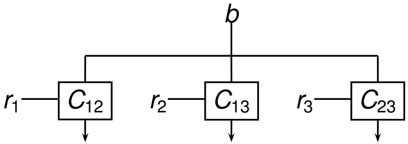
where C_{ij} is



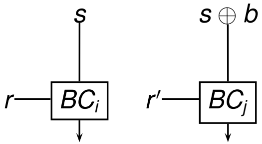
$$C^{BC_1,BC_2,BC_3}(b,r) = C_B^{C_{12},C_{13},C_{23}}(b,r) \quad \text{where} \quad C_{ij} = C_H^{BC_i,BC_j}$$

If two of the BC_i are secure, all C_{ij} are hiding, and one is also binding.

Hiding & Binding/ Hiding/ None.



where C_{ij} is



$$C^{BC_1,BC_2,BC_3}(b,r) = C_B^{C_{12},C_{13},C_{23}}(b,r) \quad \text{where} \quad C_{ij} = C_H^{BC_i,BC_j}$$

If two of the BC_i are secure, all C_{ij} are hiding, and one is also binding.
Hiding & Binding/ Hiding/ None.

- $C_B^{C_{12},C_{13},C_{23}}(b,r)$ is hiding, because *all* C_{ij} are.
- $C_B^{C_{12},C_{13},C_{23}}(b,r)$ is binding, because C_B is a $(1,3)$ robust combiner for the binding property.

Part 2

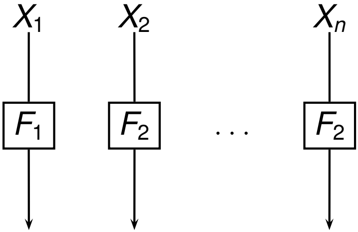
$(1, n)$-Combiners from $(1, 2)$-Combiners and Universal Schemes

D. Harnik, J.Kilian, M.Naor, O.Reingold, A.Rosen, *On Robust Combiners for Oblivious Transfer and other Primitives*, EUROCRYPT 2005

$(1, n)$ combiners from $(1, 2)$ combiners

Many robust $(1, 2)$ extend easily to $(1, n)$ combiners.

E.g. for OWFs



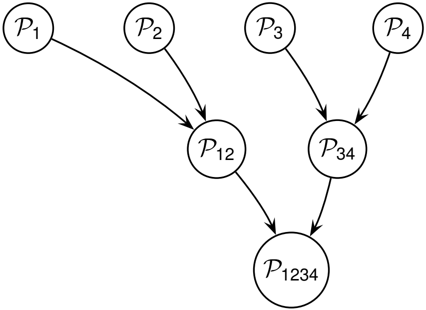
$(1, n)$ combiners from $(1, 2)$ combiners

Generic construction of a $(1, n)$ combiner $\widetilde{C}$ from a $(1, 2)$ combiner C.

$(1, n)$ combiners from $(1, 2)$ combiners

Generic construction of a $(1, n)$ combiner $\widetilde{C}$ from a $(1, 2)$ combiner C.

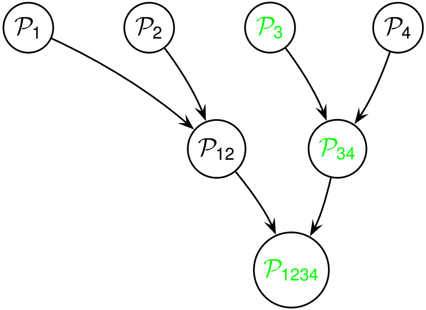
Obvious Idea: use binary tree to combiner $\mathcal{P}_1, \ldots, \mathcal{P}_{2^t}$.
$\mathcal{P}_{ij} = C^{\mathcal{P}_i, \mathcal{P}_j}$.



$(1, n)$ combiners from $(1, 2)$ combiners

Generic construction of a $(1, n)$ combiner $\widetilde{C}$ from a $(1, 2)$ combiner C.

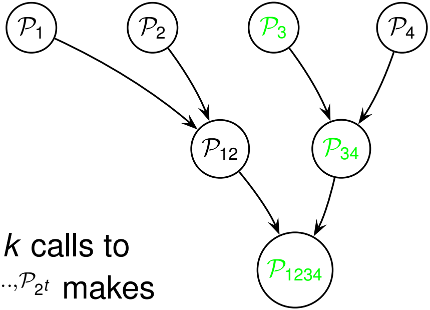
Obvious Idea: use binary tree to combiner $\mathcal{P}_1, \ldots, \mathcal{P}_{2^t}$.
$\mathcal{P}_{ij} = C^{\mathcal{P}_i, \mathcal{P}_j}$.



$(1, n)$ combiners from $(1, 2)$ combiners

Generic construction of a $(1, n)$ combiner $\widetilde{C}$ from a $(1, 2)$ combiner C.

Obvious Idea: use binary tree to combiner $\mathcal{P}_1, \ldots, \mathcal{P}_{2^t}$.
$\mathcal{P}_{ij} = C^{\mathcal{P}_i, \mathcal{P}_j}$.



Efficiency: If $C^{\mathcal{P}, \mathcal{P}'}$ makes k calls to its components, then $\widetilde{C}^{\mathcal{P}_1, \ldots, \mathcal{P}_{2^t}}$ makes k^t calls.

$(1, n)$ combiners from $(1, 2)$ combiners

A robust $(1, 2)$ *combiner is very efficient, if it calls its components at most a constant number of times.*

Lemma (HKNRR05)

If C is a very efficient robust $(1, 2)$ *combiner, then $\widetilde{C}$ is a robust* $(1, n)$ *combiner.*

If C calls each of its components k times, then $\widetilde{C}$ calls each of the components $k^{\log(n)} = poly(n)$ times.

$(1, n)$ combiners from $(1, 2)$ combiners

A robust $(1, 2)$ *combiner is very efficient, if it calls its components at most a constant number of times.*

Lemma (HKNRR05)

If C is a very efficient robust $(1, 2)$ *combiner, then $\widetilde{C}$ is a robust* $(1, n)$ *combiner.*

If C calls each of its components k times, then $\widetilde{C}$ calls each of the components $k^{\log(n)} = poly(n)$ times.

Thus for all primitives considered so far, robust $(1, n)$ combiners exist... except for BC.

$(1, n)$ combiners from $(1, 2)$ combiners for BC

For bit commitment

- Very inefficient $(1, 2)$ combiners exist via the reduction to OWFs.
- Very efficient $(2, 3)$ combiners exist (the combiner calls its components 6 times).

$(1, n)$ combiners from $(1, 2)$ combiners for BC

For bit commitment

- Very inefficient $(1, 2)$ combiners exist via the reduction to OWFs.
- Very efficient $(2, 3)$ combiners exist (the combiner calls its components 6 times).

Lemma (HKNRR05)

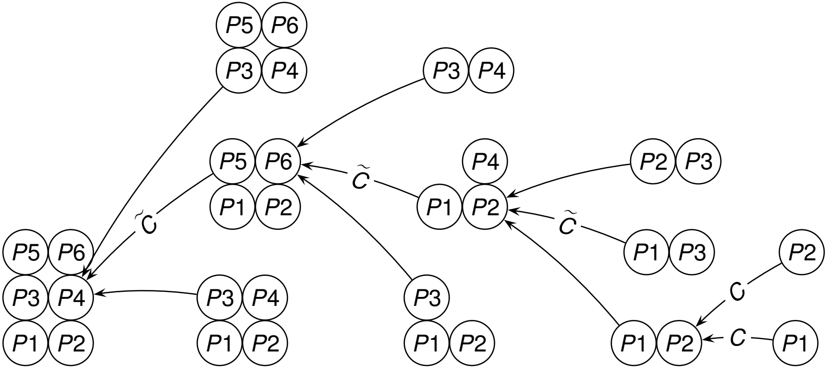
If there exists a robust $(1, 2)$ *combiner for* $\mathcal{P}$*, and a very efficient* $(2, 3)$ *combiner, then a robust* $(1, n)$ *combiner for* $\mathcal{P}$ *exists.*

$(1, n)$ combiners from $(1, 2)$ combiners for BC

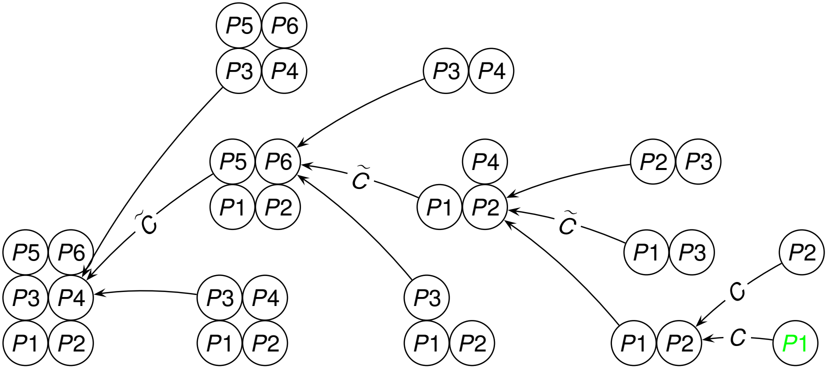
Construction of a robust $(1, k)$ combiner $\widehat{C}$ from a very efficient $(2, 3)$ combiner $\widetilde{C}$ and a $(1, 2)$ combiner C.

- If $k = 2$ use the $(1, 2)$ combiner C.
- If $k > 2$, divide k candidates into 3 groups such that each candidate is in at least 2 groups of size $2k/3$. Invoke $\widehat{C}$ recursively on each group and use $\widetilde{C}$ to combine the three groups.

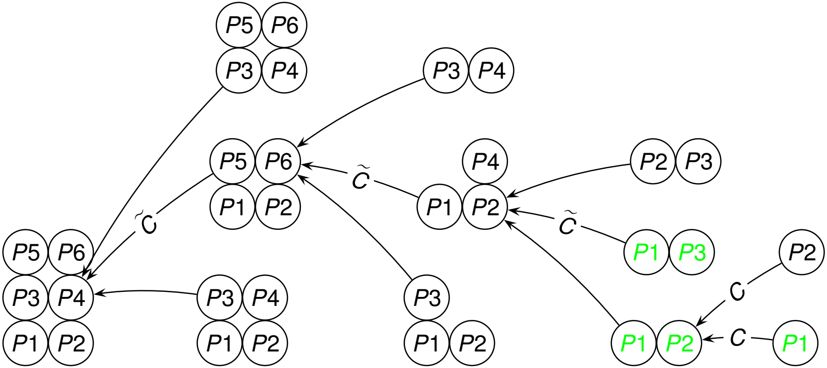
$(1, n)$ combiner $\widehat{C}$ from very efficient $(2, 3)$ combiner $\widetilde{C}$ and $(1, 2)$ combiner C



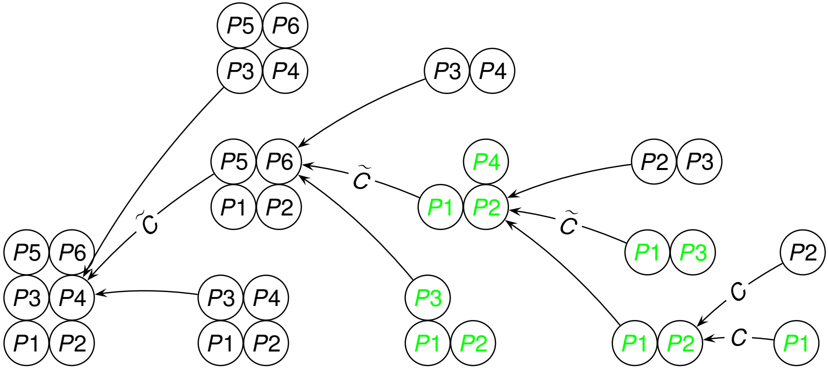
$(1, n)$ combiner $\widehat{C}$ from very efficient $(2, 3)$ combiner $\widetilde{C}$ and $(1, 2)$ combiner C



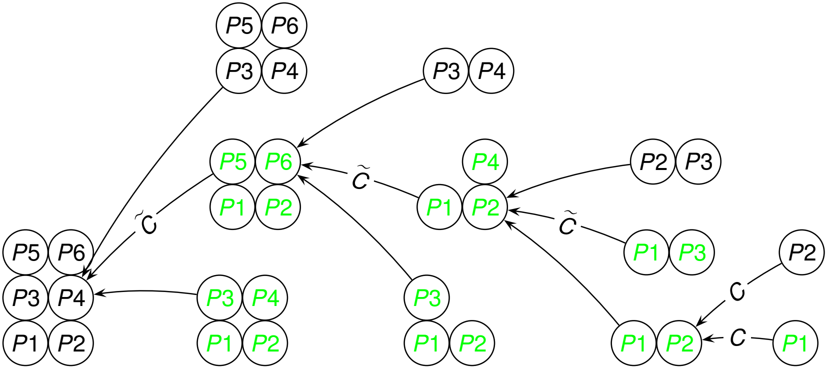
$(1, n)$ combiner $\widehat{C}$ from very efficient $(2, 3)$ combiner $\widetilde{C}$ and $(1, 2)$ combiner C



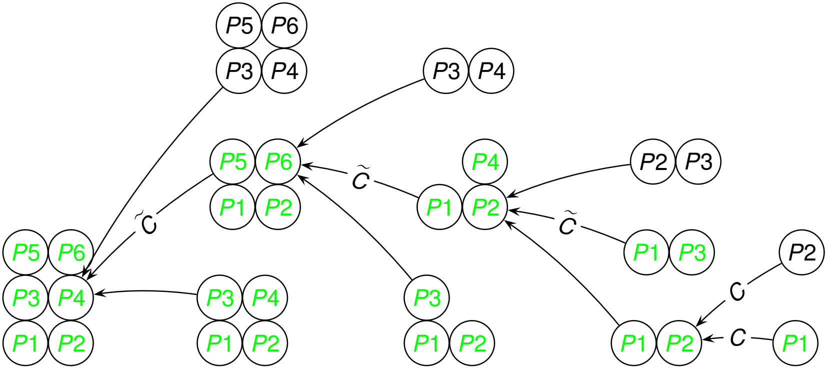
$(1, n)$ combiner $\widehat{C}$ from very efficient $(2, 3)$ combiner $\widetilde{C}$ and $(1, 2)$ combiner C



$(1, n)$ combiner $\widehat{C}$ from very efficient $(2, 3)$ combiner $\widetilde{C}$ and $(1, 2)$ combiner C



$(1, n)$ combiner $\widehat{C}$ from very efficient $(2, 3)$ combiner $\widetilde{C}$ and $(1, 2)$ combiner C



Efficiency of $\widehat{C}$

Let $t(k)$ denote the running time of $CCC^{P_1,\ldots,P_k}$, where each P_i runs in time $poly(n)$.

- $t(2) = n^d$ for some $d > 0$, as $\widehat{C}^{P_1,P_2} = C^{P_1,P_2}$.
- $t(k) = 3c \cdot t(\frac{2k}{3})$.

Where c is the number of calls that $\widetilde{C}^{P_1,P_2,P_3}$ makes to its components (e.g. $c = 6$ for the BC combiner).
Solving the recursion gives:

$$t(k) = (3c)^{\log_{3/2} k} \cdot n^d$$

This is polynomial in n for $k = poly(n)$.

Universal Schemes

Definition

A universal scheme $\mathcal{U}$ for a cryptographic primitive $\mathcal{P}$ is an explicit construction with the property that if the primitive $\mathcal{P}$ exists, then $\mathcal{U}$ is a secure implementation of $\mathcal{P}$.

Universal Schemes

Definition

A universal scheme $\mathcal{U}$ for a cryptographic primitive $\mathcal{P}$ is an explicit construction with the property that if the primitive $\mathcal{P}$ exists, then $\mathcal{U}$ is a secure implementation of $\mathcal{P}$.

Levin [Combinatorica'87] gave a universal scheme $\mathcal{U}$ for OWFs, which on input $x \in \{0,1\}^{n^2}$ is defined as

$$\mathcal{U}(x_1 \| \ldots \| x_n) = M_1[x_1] \| \ldots \| M_n[x_n]$$

- M_i is the i'th Turing Machine.
- $M_i[x]$ is the output of M_i on input x, where we stop after at most $|x|^2$ steps.

Universal Schemes

$$\mathcal{U}(x_1 \| \dots \| x_n) = M_1[x_1] \| \dots \| M_n[x_n]$$

$M_i[x]$: output of i'th TM after $|x|^2$ steps.

Universal Schemes

$$\mathcal{U}(x_1 \| \ldots \| x_n) = M_1[x_1] \| \ldots \| M_n[x_n]$$

$M_i[x]$: output of i'th TM after $|x|^2$ steps.

Efficiency: $\mathcal{U}(x_1 \| \ldots \| x_n)$ runs in time n^3.

Universal Schemes

$$\mathcal{U}(x_1\|\ldots\|x_n) = M_1[x_1]\|\ldots\|M_n[x_n]$$

$M_i[x]$: output of i'th TM after $|x|^2$ steps.

Efficiency: $\mathcal{U}(x_1\|\ldots\|x_n)$ runs in time n^3.

Hard to Invert (if OWFs exist)

- Assume *OWF* exist, then there exist *OWF*'s which run in quadratic time (use padding).
- If TM $M_m[.]$ is a OWF which runs in quadratic time, then $\mathcal{U}$ is at least as hard to invert on inputs of length $n \geq m^2$ as $M_m[.]$ on inputs of length n.

Universal Schemes

$$\mathcal{U}(x_1\|\ldots\|x_n) = M_1[x_1]\|\ldots\|M_n[x_n]$$

$M_i[x]$: output of i'th TM after $|x|^2$ steps.

Efficiency: $\mathcal{U}(x_1\|\ldots\|x_n)$ runs in time n^3.

Hard to Invert (if OWFs exist)

- Assume *OWF* exist, then there exist *OWF*'s which run in quadratic time (use padding).
- If TM $M_m[.]$ is a OWF which runs in quadratic time, then $\mathcal{U}$ is at least as hard to invert on inputs of length $n \geq m^2$ as $M_m[.]$ on inputs of length n.

Because

$$C^{f_1,\ldots,f_n}(x_1\|\ldots\|x_n) = f_1(x)\|\ldots\|f_n(x_n)$$

is a robust $(1, n)$ combiner for *OWFs*.

Universal Schemes

Lemma (HKNRR05)

For any primitive $\mathcal{P}$, if:

1. *We know a polynomial $p(.)$ s.t. if $\mathcal{P}$ exists, there exists an implementation which runs in time $p(n)$.*
2. *We have a $(1, n)$ robust combiner for $\mathcal{P}$.*

Then we can provide a Universal scheme for $\mathcal{P}$

Universal Schemes

Lemma (HKNRR05)

For any primitive $\mathcal{P}$, if:

1. *We know a polynomial $p(.)$ s.t. if $\mathcal{P}$ exists, there exists an implementation which runs in time $p(n)$.*
2. *We have a $(1, n)$ robust combiner for $\mathcal{P}$.*

Then we can provide a Universal scheme for $\mathcal{P}$

Universal schemes for all all primitives we saw so far exist!

OT Combiner???

Lemma (HKNRR05)

A very efficient $(2, 3)$ *combiner for oblivious transfer exists.*

The construction is very similar to the $(2, 3)$ BC combiner, but unlike for BC, no $(1, 2)$ combiner is known.

Open Problem

Does there exist a $(1, 2)$ *combiner for OT?*

Such a combiner would imply a $(1, n)$ combiner for OT, and further

Lemma

Any $(1, 2)$ *combiner for OT can be used to construct a universal OT-scheme.*

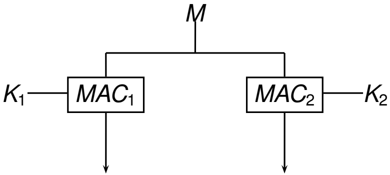
Combiners for Collision Resistance

D.Boneh, X.Boyen: *On the Impossibility of Efficiently Combining Collision Resistant Hash Functions.* CRYPTO 2006

K.Pietrzak: *Non-trivial Black-Box Combiners for Collision-Resistant Hash-Functions Don't Exist.* EUROCRYPT 2007

R.Canetti, R.Rivest, M.Sudan, L.Trevisan, S.Vadhan, H.Wee: *Amplifying Collision Resistance: A Complexity-Theoretic Treatment.* CRYPTO 2007

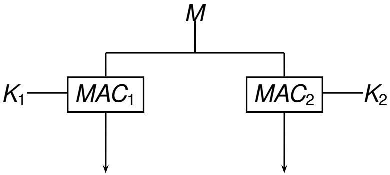
MAC Combiner Revisited



$$C^{MAC_1,MAC_2}([K_1,K_2],M) = MAC_1(K_1,M) \| MAC_2(K_2,M)$$

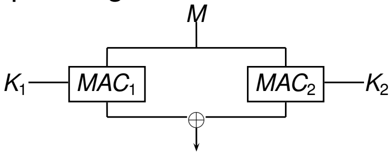
Unfortunately output length is doubled...

MAC Combiner Revisited



$$C^{MAC_1,MAC_2}([K_1,K_2],M) = MAC_1(K_1,M) \| MAC_2(K_2,M)$$

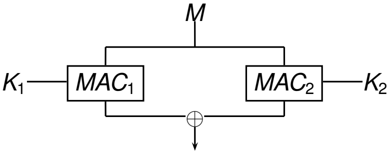
Unfortunately output length is doubled...



$$C^{MAC_1,MAC_2}([K_1,K_2],M) = MAC_1(K_1,M) \oplus MAC_2(K_2,M)$$

One can XOR the outputs, and the combiner stays robust!

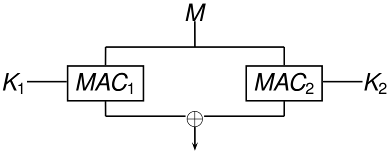
MAC Combiner Revisited



$$C^{MAC_1, MAC_2}([K_1, K_2], M) = MAC_1(K_1, M) \oplus MAC_2(K_2, M)$$

is a robust combiner:

MAC Combiner Revisited

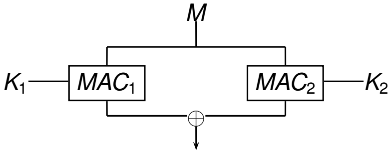


$$C^{MAC_1,MAC_2}([K_1,K_2],M) = MAC_1(K_1,M) \oplus MAC_2(K_2,M)$$

is a robust combiner:

- Assume $A^{C^{MAC_1,MAC_2}([K_1,K_2],.)}$ outputs forgery with non-negligible probability.

MAC Combiner Revisited

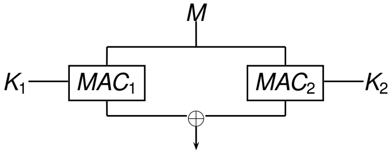


$$C^{MAC_1, MAC_2}([K_1, K_2], M) = MAC_1(K_1, M) \oplus MAC_2(K_2, M)$$

is a robust combiner:

- Assume $A^{C^{MAC_1, MAC_2}([K_1, K_2], .)}$ outputs forgery with non-negligible probability.
- To break $MAC_1(K, .)$: Sample key K' for MAC_2.

MAC Combiner Revisited

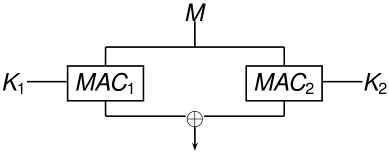


$$C^{MAC_1,MAC_2}([K_1,K_2],M) = MAC_1(K_1,M) \oplus MAC_2(K_2,M)$$

is a robust combiner:

- Assume $A^{C^{MAC_1,MAC_2}([K_1,K_2],.)}$ outputs forgery with non-negligible probability.
- To break $MAC_1(K,.)$: Sample key K' for MAC_2.
- Let A attack $MAC_1(K,.) \oplus MAC_2(K',.)$.

MAC Combiner Revisited

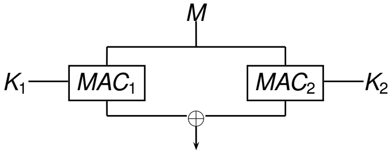


$$C^{MAC_1,MAC_2}([K_1, K_2], M) = MAC_1(K_1, M) \oplus MAC_2(K_2, M)$$

is a robust combiner:

- Assume $A^{C^{MAC_1,MAC_2}([K_1,K_2],.)}$ outputs forgery with non-negligible probability.
- To break $MAC_1(K, .)$: Sample key K' for MAC_2.
- Let A attack $MAC_1(K, .) \oplus MAC_2(K', .)$.
- A outputs forgery (M, ϕ) for $C^{MAC_1,MAC_2}([K, K'], .)$

MAC Combiner Revisited



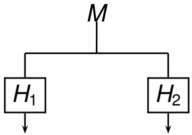
$$C^{MAC_1,MAC_2}([K_1,K_2],M) = MAC_1(K_1,M) \oplus MAC_2(K_2,M)$$

is a robust combiner:

- Assume $A^{C^{MAC_1,MAC_2}([K_1,K_2],.)}$ outputs forgery with non-negligible probability.
- To break $MAC_1(K,.)$: Sample key K' for MAC_2.
- Let A attack $MAC_1(K,.) \oplus MAC_2(K',.)$.
- A outputs forgery (M,ϕ) for $C^{MAC_1,MAC_2}([K,K'],.)$
- Output forgery (M,ϕ') for $MAC_1(K,.)$, where

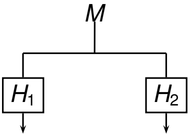
$$\phi' = \phi \oplus MAC_2(K,M))$$

CRHF Combiner Revisited



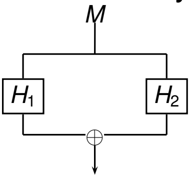
$$C^{H_1,H_2}(M) = H_1(M) \| H_2(M)$$

CRHF Combiner Revisited



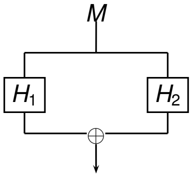
$$C^{H_1,H_2}(M) = H_1(M) \| H_2(M)$$

Output length doubled, unfortunately (unlike for MACs)



is not robust.

CRHF Combiner Revisited



$$C^{H_1,H_2}([K_1,K_2],M) = H_1(K_1,M) \oplus H_2(K_2,M)$$

Is not robust. Let $H_1, H_2 : \{0,1\}^m \to \{0,1\}^n \setminus \{A,B\}$ be CRHFs. For all keys K and any $X, Y \in \{0,1\}^m$ redefine

$$H_1(K,X) = A \qquad H_1(K,Y) = B$$

$$H_2(K,X) = B \qquad H_2(K,Y) = A$$

Then the inputs X and Y collide in C^{H_1,H_2}:

$$\begin{aligned} C^{H_1,H_2}([K_1,K_2],X) &= A \oplus A = 0^n \\ C^{H_1,H_2}([K_1,K_2],Y) &= B \oplus B = 0^n \end{aligned}$$