

CSE 191, Sp08, Bellare

- Small number of students (1–8)
- Projects / research
- Implementation or theoretical
- Weekly one-on-one meetings, no lectures
- Course grade based on project report

More crypto and security

CSE 207, Fa08, Bellare

- Graduate introduction to cryptography
- Focus on provable-security
- Material overlaps with 107 but changes in focus as above

CSE 127, Savage / Shacham

- Undergraduate introduction to security
- Systems issues

CSE 227, Savage/Shacham

- Graduate introduction to security

CSE 208 is a “Topics” course whose content varies from year to year.
Some past topics:

- Pairing-based cryptography
- Lattices in cryptography
- Zero-knowledge
- Electronic payments

Adam O'Neill

- CSE 107, 191, 207
- Now in PhD program at Georgia Tech
- Producing lots of good research!

Darrel Carbajal

- CSE 107, 191, 207, 208
- Now in PhD program at UCLA

Grad school?

- MS: 2 years
- PhD: 5+ years, research-orientated

You can consider a PhD if you enjoy research: solving new problems, exploring the unknown.

An MS is to gain expertise.

PhD application process

- Applications for Fall due in previous winter
- PhD students are usually fully funded through fellowships, RA-ships and TA-ships.
- Admission to top schools is very competitive but there are niche schools that are excellent in specific areas
- To get admitted to a good program you need a high GPA, strong recommendation letters and a convincing statement of purpose.
- Best of all: accomplished research!

APPLICATIONS AND PROTOCOLS

Agenda

- Internet Casino
 - Commitment
 - Shared coin flips
 - Threshold cryptography
 - Forward security
 - Identity-based encryption
 - Searchable encryption
 - Certified e-mail
- Zero-knowledge
 - Oblivious transfer
 - Secure computation
 - Group signatures
 - Multi-signatures
 - Aggregate signatures
 - Electronic voting
 - Auctions

Internet Casino: Protocol G1

Player

Casino

$\xrightarrow{\$1, G}$

$T \xleftarrow{\$} \{1, 2, \dots, 100\}$
if $G = T$ **then** $d \leftarrow \$200$
else $d \leftarrow \$0$

$\xleftarrow{T, d}$

Would you play?

Internet Casino: Protocol G1

Player

Casino

$\xrightarrow{\$1, G}$

$T \xleftarrow{\$} \{1, 2, \dots, 100\}$

if $G = T$ **then** $d \leftarrow \$200$

else $d \leftarrow \$0$

$\xleftarrow{T, d}$

Would you play?

Expected value of d is $\$200(\frac{1}{100}) = \$2 > \$1$ so probability theory says that the player will earn money by playing.

Internet Casino: Protocol G1 problem

Problem: Casino can cheat:

Player

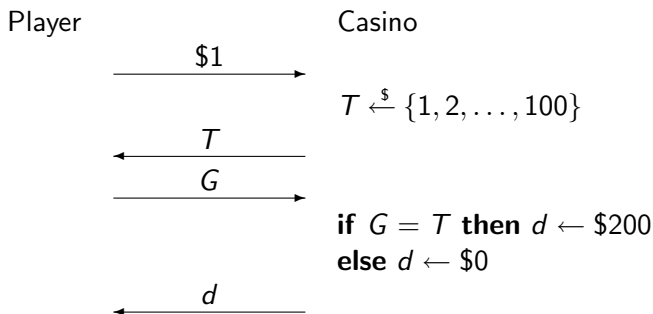
Casino

$\$1, G$

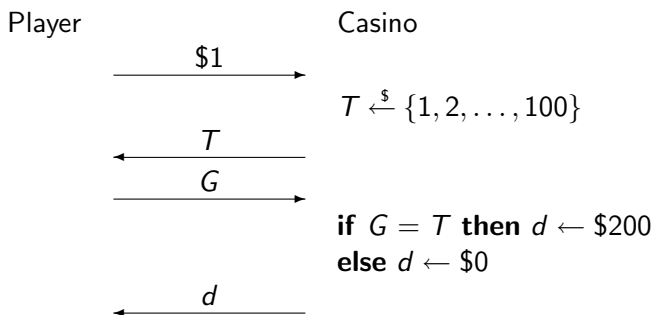
$T \xleftarrow{\$} \{1, 2, \dots, 100\} \setminus \{G\}$
 $d \leftarrow \$0$

T, d

Internet Casino: Protocol G2



Internet Casino: Protocol G2



But now player can always win by setting $G = T$. No casino would do this!

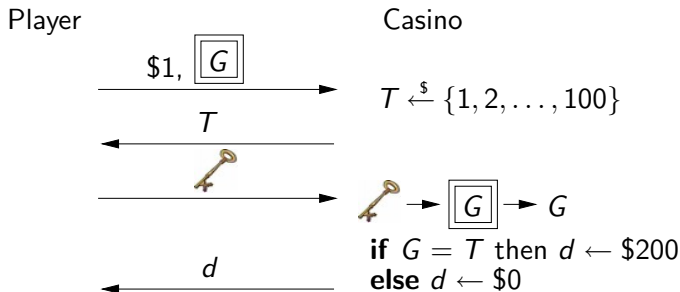
Internet Casino problem

Player and Casino need to exchange G , T so that

- Casino cannot choose T as a function of G .
- Player cannot choose G as a function of T .

How do we resolve this Catch-22 situation?

"Internet" Casino: Protocol G3

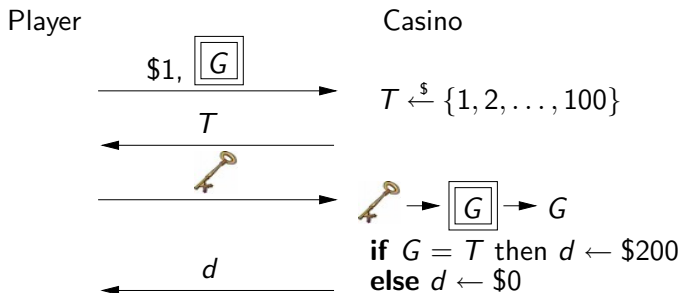


is a locked safe containing a piece of paper with G written on it.



is a key to open the safe.

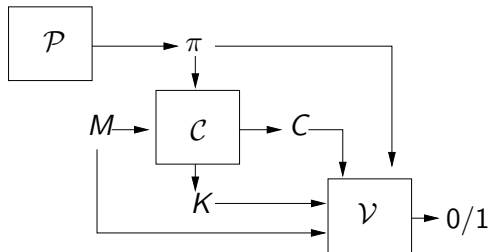
"Internet" Casino: Protocol G3





- Casino cannot choose T as a function of G because, without the key, it cannot see G .
- Player cannot choose G as a function of T because, by putting it in the safe, she is committed to it in the first move.

Commitment Schemes

A commitment scheme $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ is a triple of algorithms

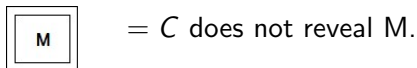


Parameter generation algorithm \mathcal{P} is run once by a trusted party to produce public parameters π .

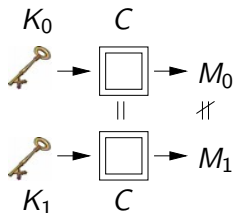
		In Internet Casino
M	Data being committed	G
C	Committed	
K	Decommitment key	

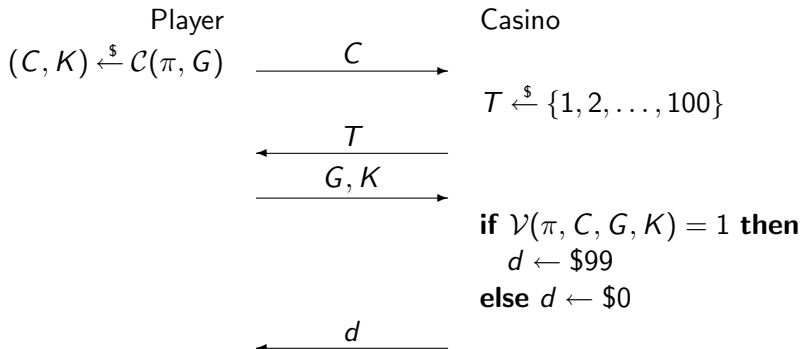
Security properties

- Hiding: A comital C generated via $(C, K) \xleftarrow{\$} \mathcal{C}(\pi, M)$ should not reveal information about M .



- Binding: It should be hard to find C, M_0, M_1, K_0, K_1 such that $M_0 \neq M_1$ but $\mathcal{V}(\pi, C, M_0, K_0) = \mathcal{V}(\pi, C, M_1, K_1) = 1$.





Hiding Formally

Let $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ be a commitment scheme and A an adversary.

Game $\text{HIDE}_{\mathcal{CS}}$

procedure Initialize

$\pi \xleftarrow{\$} \mathcal{P}; b \xleftarrow{\$} \{0, 1\}$

return π

procedure $\text{LR}(M_0, M_1)$

$(C, K) \xleftarrow{\$} \mathcal{C}(\pi, M_b)$

return C

procedure Finalize(b')

return $(b = b')$

The hiding-advantage of A is

$$\text{Adv}_{\mathcal{CS}}^{\text{hide}}(A) = 2 \cdot \Pr \left[\text{HIDE}_{\mathcal{CS}}^A \Rightarrow \text{true} \right] - 1.$$

Binding Formally

Let $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ be a commitment scheme and A an adversary.

Game $\text{BIND}_{\mathcal{CS}}$

procedure Initialize

$\pi \xleftarrow{\$} \mathcal{P}$

return π

procedure Finalize(C, M_0, M_1, K_0, K_1)

$v_0 \leftarrow \mathcal{V}(\pi, C, M_0, K_0)$

$v_1 \leftarrow \mathcal{V}(\pi, C, M_1, K_1)$

return ($v_0 = v_1 = 1$ and $M_0 \neq M_1$)

The binding-advantage of A is

$$\text{Adv}_{\mathcal{CS}}^{\text{bind}}(A) = \Pr \left[\text{BIND}_{\mathcal{CS}}^A \Rightarrow \text{true} \right].$$

Commitment from symmetric encryption

Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ be the commitment scheme where \mathcal{P} returns $\pi = \varepsilon$ and

Algorithm $\mathcal{C}(\pi, M)$

$K \xleftarrow{\$} \mathcal{K}; C \xleftarrow{\$} \mathcal{E}_K(M)$

return (C, K)

Algorithm $\mathcal{V}(\pi, C, M, K)$

if $\mathcal{D}_K(C) = M$ **then return** 1

else return 0

Is this secure?

Commitment from symmetric encryption

Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ be the commitment scheme where \mathcal{P} returns $\pi = \varepsilon$ and

<p>Algorithm $\mathcal{C}(\pi, M)$ $K \xleftarrow{\\$} \mathcal{K}; C \xleftarrow{\\$} \mathcal{E}_K(M)$ return (C, K)</p>	<p>Algorithm $\mathcal{V}(\pi, C, M, K)$ if $\mathcal{D}_K(C) = M$ then return 1 else return 0</p>
--	---

Is this secure?

- It is certainly hiding.

Commitment from symmetric encryption

Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ be the commitment scheme where \mathcal{P} returns $\pi = \varepsilon$ and

<p>Algorithm $\mathcal{C}(\pi, M)$ $K \xleftarrow{\\$} \mathcal{K}; C \xleftarrow{\\$} \mathcal{E}_K(M)$ return (C, K)</p>	<p>Algorithm $\mathcal{V}(\pi, C, M, K)$ if $\mathcal{D}_K(C) = M$ then return 1 else return 0</p>
--	---

Is this secure?

- It is certainly hiding.
- But need not be binding: it may be possible to find C, M_0, M_1, K_0, K_1 such that

$$\mathcal{D}_{K_0}(C) = M_0 \text{ and } \mathcal{D}_{K_1}(C) = M_1$$

For example this is easy when \mathcal{SE} is *CBC* encryption.

Commitment from public key encryption

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme and let $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ be the commitment scheme where

Alg \mathcal{P} $(pk, sk) \xleftarrow{\$} \mathcal{K}$ $\pi \leftarrow pk$ return π	Alg $\mathcal{C}(pk, M)$ $K \xleftarrow{\$} \{0, 1\}^k$ $C \leftarrow \mathcal{E}_{pk}(M; K)$ return (C, K)	Alg $\mathcal{V}(pk, C, M, K)$ if $\mathcal{E}_{pk}(M; K) = C$ then return 1 else return 0
--	--	---

- $\mathcal{E}_{pk}(M; K)$ means encryption of M with coins K .

Commitment from public key encryption

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme and let $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ be the commitment scheme where

Alg \mathcal{P} $(pk, sk) \xleftarrow{\$} \mathcal{K}$ $\pi \leftarrow pk$ return π	Alg $\mathcal{C}(pk, M)$ $K \xleftarrow{\$} \{0, 1\}^k$ $C \leftarrow \mathcal{E}_{pk}(M; K)$ return (C, K)	Alg $\mathcal{V}(pk, C, M, K)$ if $\mathcal{E}_{pk}(M; K) = C$ then return 1 else return 0
---	---	---

- $\mathcal{E}_{pk}(M; K)$ means encryption of M with coins K .
- Certainly hiding.

Commitment from public key encryption

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme and let $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ be the commitment scheme where

Alg \mathcal{P}	Alg $\mathcal{C}(pk, M)$	Alg $\mathcal{V}(pk, C, M, K)$
$(pk, sk) \xleftarrow{\$} \mathcal{K}$	$K \xleftarrow{\$} \{0, 1\}^k$	if $\mathcal{E}_{pk}(M; K) = C$ then
$\pi \leftarrow pk$	$C \leftarrow \mathcal{E}_{pk}(M; K)$	return 1
return π	return (C, K)	else return 0

- $\mathcal{E}_{pk}(M; K)$ means encryption of M with coins K .
- Certainly hiding.
- Binding too since C has only one decryption relative to pk , namely $M = \mathcal{D}_{sk}(C)$.

Commitment from hashing

Let H be a hash function and $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ the commitment scheme where \mathcal{P} returns $\pi = \varepsilon$ and

$\text{Alg } \mathcal{C}(\pi, M)$	
$C \leftarrow H(M)$	$\text{Alg } \mathcal{V}(\pi, C, M, K)$
$K \leftarrow M$	return ($C = H(M)$ and $M = K$)
return (C, K)	

This is

Commitment from hashing

Let H be a hash function and $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ the commitment scheme where \mathcal{P} returns $\pi = \varepsilon$ and

$\text{Alg } \mathcal{C}(\pi, M)$	
$C \leftarrow H(M)$	$\text{Alg } \mathcal{V}(\pi, C, M, K)$
$K \leftarrow M$	return ($C = H(M)$ and $M = K$)
return (C, K)	

This is

- Binding if H is collision-resistant.

Commitment from hashing

Let H be a hash function and $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ the commitment scheme where \mathcal{P} returns $\pi = \varepsilon$ and

Alg $\mathcal{C}(\pi, M)$		Alg $\mathcal{V}(\pi, C, M, K)$
$C \leftarrow H(M)$		return ($C = H(M)$ and $M = K$)
$K \leftarrow M$		
return (C, K)		

This is

- Binding if H is collision-resistant.
- But not hiding.

For example in the Internet Casino $M = G \in \{1, \dots, 100\}$ so given $C = H(M)$ the casino can recover M via

```
for  $i = 1, \dots, 100$  do  
  if  $H(i) = C$  then return  $i$ 
```

Commitment from hashing

A better scheme is $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ where \mathcal{P} returns $\pi = \varepsilon$ and

Alg $\mathcal{C}(\pi, M)$

$K \xleftarrow{\$} \{0, 1\}^{128}$

$C \leftarrow H(K || M)$

return (C, K)

Alg $\mathcal{V}^H(\pi, C, M, K)$

return $(H(K || M) = C)$

Commitment schemes usage

Commitment schemes are very broadly and widely used across all kinds of protocol design and in particular to construct zero-knowledge proofs.

Flipping a common coin

- Alice and Bob are getting divorced
- They want to decide who keeps the Lexus
- They agree to flip a coin, but
- Alice is in NY and Bob is in LA

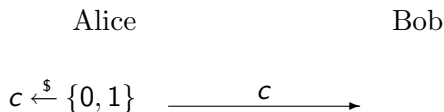
Protocol CF1:



Flipping a common coin

- Alice and Bob are getting divorced
- They want to decide who keeps the Lexus
- They agree to flip a coin, but
- Alice is in NY and Bob is in LA

Protocol CF1:

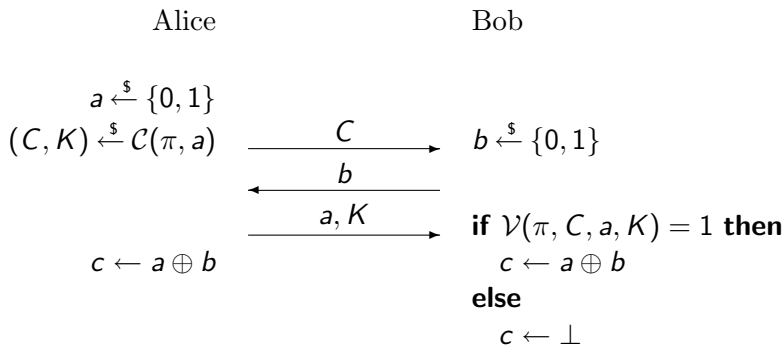


Bob is not too smart but he doesn't like it...

Can you help them out?

Protocol CF2

Let $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ be a commitment scheme.



c is the common coin. Neither party can control it.

Key Exposure

- Cryptography (whether symmetric or asymmetric) relies on secret keys.
- The secret keys need to be stored on some system.
- Systems are vulnerable to compromise via breakin due to viruses, worms or OS holes.
- This puts secret keys at risk of exposure.

Key exposure due to system compromise is a large, important and immediate threat.

What can we do about it?

Protecting against key exposure

- Distribution (sharing) — Threshold signatures.
- Key evolution — Forward security.

Threshold signatures

Suppose a CA has RSA public verification key N, e and corresponding secret signing key N, d . Exposure of d would have devastating effects, invalidating millions of certificates.

In a threshold signature scheme

- d is split into “shares” d_1, \dots, d_n , each stored on a different server.
- Possession of t or less shares does not permit recovery of d , meaning the system can tolerate t compromised servers.
- Signatures are created via a distributed computation in which d is never re-constituted in any one place at any one time.

Threshold RSA signatures

Signer keys: $pk = N, e$ and $sk = N, d$.

Signature of M is $\mathcal{S}_{N,d}(M) = H(M)^d \bmod N$.

Let $P = \varphi(N)$.

Splitting the key: $d \rightarrow d_1, d_2, d_3, d_4, d_5$ such that d_1, \dots, d_5 are uniformly distributed in Z_P^* subject to

$$d_1 + d_2 + d_3 + d_4 + d_5 \equiv d \pmod{P}$$

How?

Let $d_1, d_2, d_3, d_4 \xleftarrow{\$} Z_m^*$ and let $d_5 \leftarrow d - (d_1 + d_2 + d_3 + d_4) \bmod P$.

Store d_i on server i ($1 \leq i \leq 5$) and delete d .

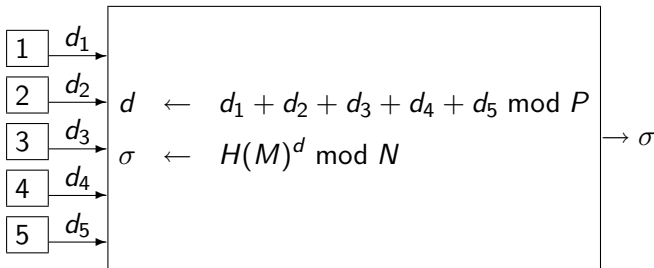
Given N, e and any $t = 4$ or less of the 5 shares d_1, \dots, d_5 , it is hard to figure out d . More precisely it is just as hard as $N, e \rightarrow d$.

This is an instance of $n - 1$ out of n secret sharing with $n = 5$.

Naive threshold signing

To sign M

- Server i sends d_i to a Combiner.
- Combiner computes



Problems

- d is present on Combiner system, even if ephemerally, and can be exposed.
- Combiner needs to store $P = \varphi(N)$, but exposure of P enables recovery of d .

Threshold signing

If $d \equiv d_1 + d_2 + d_3 + d_4 + d_5 \pmod{P}$ then modulo N we have

$$\begin{aligned} H(M)^d &\equiv H(M)^{(d_1+d_2+d_3+d_4+d_5) \bmod P} \\ &\equiv H(M)^{d_1} H(M)^{d_2} H(M)^{d_3} H(M)^{d_4} H(M)^{d_5} \end{aligned}$$

so to sign M :

$$M \rightarrow \boxed{1} \xrightarrow{\sigma_1 = H(M)^{d_1} \bmod N}$$

$$M \rightarrow \boxed{2} \xrightarrow{\sigma_2 = H(M)^{d_2} \bmod N}$$

$$M \rightarrow \boxed{3} \xrightarrow{\sigma_3 = H(M)^{d_3} \bmod N}$$

$$M \rightarrow \boxed{4} \xrightarrow{\sigma_4 = H(M)^{d_4} \bmod N}$$

$$M \rightarrow \boxed{5} \xrightarrow{\sigma_5 = H(M)^{d_5} \bmod N}$$

$$\sigma \leftarrow \sigma_1 \sigma_2 \sigma_3 \sigma_4 \sigma_5 \bmod N \rightarrow \sigma$$

Security of threshold scheme

Let the adversary

- Get d_i for all $i \in T$ where $T \subseteq \{1, \dots, 5\}$ is a set of size 4 or less of adversary's choice.
- See all communications from servers to Combiner.
- Mount a chosen-message attack.

Still, forging the signature of a new message is as hard as breaking UF-CMA security of the signature scheme directly.

Issues with threshold signatures

- Common-mode failures: Compromising 5 systems may not be much harder than compromising 1 if they all run the same OS.
- A CA may have the resources to do threshold signatures but an average user may not.

Price of revocation

November 1: Alice uses her secret signing key sk to produce a signature σ of a document M .

- (M, σ) could be a certificate issued by CA Alice.
- M could be a contract.
- M could be a check payable to Bob.

December 1: sk is exposed and Alice revokes the corresponding public key pk .

December 2: Bob receives (M, σ) . Then Bob will reject it as unauthentic since pk has been revoked.

Indeed, he should, since he has no way to tell that σ was produced by Alice and not (later) by the adversary.

This is a major nuisance ...

Time stamping does not solve this

November 1: Alice uses the secret signing key sk to produce a signature σ on $11/1/07||M$.

December 1: sk is exposed and corresponding pk is revoked.

December 2: Bob receives $(11/1/07||M, \sigma)$ and knows pk was revoked only on 12/1/07. So should he accept $(11/1/07||M, \sigma)$?

Time stamping does not solve this

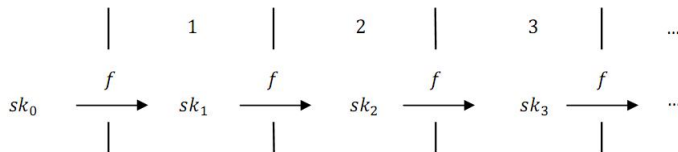
November 1: Alice uses the secret signing key sk to produce a signature σ on $11/1/07||M$.

December 1: sk is exposed and corresponding pk is revoked.

December 2: Bob receives $(11/1/07||M, \sigma)$ and knows pk was revoked only on 12/1/07. So should he accept $(11/1/07||M, \sigma)$?

NO! Adversary obtaining sk on 12/1/07 could create a signature on $mm/dd/yy || M$ for any $mm/dd/yy$ of her choice, even past ones.

Forward security [An, BeMi99]



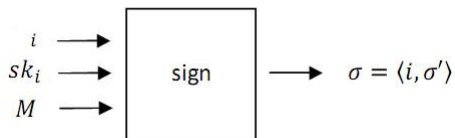
p_k : signer's public key, used to verify all the signatures.
Certified as usual

sk_i : secret signing key in time period i

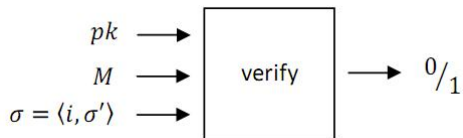
Key evolution: At the start of time period i , the signer

- updates sk_{i-1} to $sk_i = f(sk_{i-1})$
- deletes sk_{i-1} from the system

Signing and verifying



A signature always includes the time period in which it was created.



Break-in and security requirement

If adversary breaks in to system in some time period b it obtains sk_b and hence also $sk_{b+1}, sk_{b+2}, \dots$ as f is public.

Security requirement: Possession of sk_b does not allow adversary to

- Compute sk_i for $i < b$
- Forge signatures for time periods $i < b$, even after a chosen message attack

Note: this requires the update function f to be 1-way.

What this means

time period length = 1 day

time period	what happens
8 (Jan 8)	Alice produces signature $\sigma = \langle 8, \sigma' \rangle$ of document M under sk_8
32 (Feb 1)	Alice's system compromised: sk_{32}, sk_{33}, \dots exposed
33 (Feb 2)	Alice's pk revoked for period $t = 32$
42 (Feb 11)	Bob receives (M, σ)

Even though pk is revoked, Bob on Feb 11 can accept (M, σ) as Alice's signature since $\sigma = \langle 8, \sigma' \rangle$ and $8 < t = 32$. Forward security guarantees that (M, σ) is authentic.

- Idea of forward-secure signatures suggested by Anderson.
- Definitions and first schemes by [BeMi99]
- Since then, many other schemes [IR, Kr, MMM, ...]
- Extensions: key-insulation, intrusion resilience
- Extensions: forward secure encryption [CHK], forward secure symmetric cryptography (PRGs etc) [BeYe]

Secure summation

Suppose we have n parties $1, \dots, n$

Party i has an integer x_i

The parties want to know the value of

$$f(x_1, \dots, x_n) = x_1 + \dots + x_n$$

Secure summation

Suppose we have n parties $1, \dots, n$

Party i has an integer x_i

The parties want to know the value of

$$f(x_1, \dots, x_n) = x_1 + \dots + x_n$$

Easy: Let

- Party i send x_i to party 1 ($2 \leq i \leq n$)
- Party 1 computes $f(x_1, \dots, x_n) = x_1 + \dots + x_n$ and broadcasts it

Secure summation

Suppose we have n parties $1, \dots, n$

Party i has an integer x_i

The parties want to know the value of

$$f(x_1, \dots, x_n) = x_1 + \dots + x_n$$

Easy: Let

- Party i send x_i to party 1 ($2 \leq i \leq n$)
- Party 1 computes $f(x_1, \dots, x_n) = x_1 + \dots + x_n$ and broadcasts it

What they don't like about this: Party 1 now knows everyone's values

Privacy constraint: Party i does not wish to reveal x_i

Party i has input x_i ($1 \leq i \leq n$). The parties want to know $f(x_1, \dots, x_n) = x_1 + \dots + x_n$ but do not want to reveal their inputs in the process.

Scenarios:

- x_i = score of student i on midterm exam
- x_i = salary of employee i
- $x_i \in \{0, 1\}$ = vote of voter i on proposition X on ballot

The TOP solution

Suppose we had a trusted outside party TOP. Then the problem is easily solved:

- Party i sends x_i to TOP ($1 \leq i \leq n$)
- TOP computes $s = x_1 + \dots + x_n$
- TOP sends s to party i ($1 \leq i \leq n$)

But we may not want some “trusted” party to know our data (votes, scores, salaries, ...)

The model and goal

Parties i, j are connected via a secure channel ($1 \leq i, j \leq n$).

Privacy and authenticity of messages sent over channel are guaranteed.

The parties will exchange messages to arrive at $f(x_1, \dots, x_n)$.

If $i \neq j$ then, at the end of the protocol, party i should not know x_j .

Setup for secure communication protocol

Let N be such that $x_1, \dots, x_n \in Z_N = \{0, \dots, N-1\}$.

Let $M = nN$.

Let S denote $x_1 + \dots + x_n$.

We will compute $S \bmod M$, which is just S since

$$x_1 + \dots + x_n \leq n(N-1) < M$$

Protocol step 1: secret sharing

For $i = 1, \dots, n$ party i

- Picks $x_{i,1}, \dots, x_{i,n} \in Z_M$ at random subject to $x_{i,1} + \dots + x_{i,n} \equiv x_i \pmod{M}$
- Sends $x_{i,j}$ to party j over secure channel ($1 \leq j \leq n$)

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{bmatrix} \begin{array}{l} \rightarrow x_1 \\ \rightarrow x_2 \\ \rightarrow x_3 \\ \rightarrow x_4 \end{array}$$

Observation: $x_{i,j}$ is a random number unrelated to x_i so party j has no information about x_i ($i \neq j$)

Protocol step 2,3: Column sums and conclusion

$$\begin{array}{ccccccc} \left[\begin{array}{cccc} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{array} \right] & \rightarrow & x_1 \\ & & & & \rightarrow & x_2 \\ & & & & \rightarrow & x_3 \\ & & & & \rightarrow & x_4 \\ & \downarrow & \downarrow & \downarrow & \downarrow & \\ & C_1 & C_2 & C_3 & C_4 & \end{array}$$

For $j = 1, \dots, n$ party j

- Computes $C_j = x_{1,j} + x_{2,j} + \dots + x_{n,j} \bmod M$
- Sends C_j to party i ($1 \leq i \leq n$)

Observation: Modulo M we have

$$S \equiv \sum_{i=1}^n x_i \equiv \sum_{i=1}^n \sum_{j=1}^n x_{i,j} \equiv \sum_{j=1}^n \sum_{i=1}^n x_{i,j} \equiv \sum_{j=1}^n C_j$$

So each party can compute $S \leftarrow (C_1 + \dots + C_n) \bmod M$

Security of the protocol

$$\begin{array}{cccc} \left[\begin{array}{cccc} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{array} \right] & \begin{array}{l} \rightarrow x_1 \\ \rightarrow x_2 \\ \rightarrow x_3 \\ \rightarrow x_4 \end{array} \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow & \\ c_1 & c_2 & c_3 & c_4 \end{array}$$

At end of protocol, party 1 knows

- x_1 , and the first-row entries of the matrix
- The sum $S = x_1 + x_2 + x_3 + x_4$
- c_1, c_2, c_3, c_4
- The first column entries $x_{1,1}, x_{2,1}, x_{3,1}, x_{4,1}$

Claim: Party 1 has learned nothing about x_2

Suppose parties 1, 2 get together. Can they figure out something about x_3 ?

Suppose parties 1, 2 get together. Can they figure out something about x_3 ?

- Not really.

Suppose parties 1, 2 get together. Can they figure out something about x_3 ?

- Not really.

Can parties 1,2,3 figure out something about x_4 ?

Suppose parties 1, 2 get together. Can they figure out something about x_3 ?

- Not really.

Can parties 1,2,3 figure out something about x_4 ?

- Yes, because

$$x_4 = S - (x_1 + x_2 + x_3)$$

Security under collusions

At the end of the protocol, a set $T \subseteq \{1, \dots, n\}$ of parties certainly can compute

$$\sum_{j \notin T} x_j = S - \sum_{i \in T} x_i$$

However, the set T of parties learns nothing about $\{x_j : j \notin T\}$ other than this.

Secure Computation

Parties $1, \dots, n$

Party i has private input x_i

They want to compute $f(x_1, \dots, x_n)$

Fact: For any function f , there is a $n/2$ - private protocol to compute it.

A protocol is t -private if any t parties, getting together, cannot figure out anything about the input of the other parties other than implied by the value of $f(x_1, \dots, x_n)$.

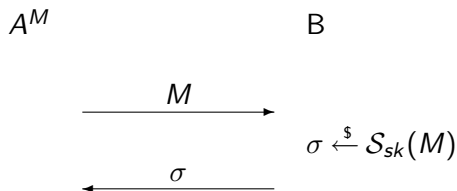
The protocol views f as a circuit (program) and computes it gate (instruction) by gate (instruction).

Enormous body of research.

Blind Signatures

Bob (B) has secret signing key sk and public verification key pk

Alice (A) wants to get Bob's signature on a document M . But she does not want to do this:

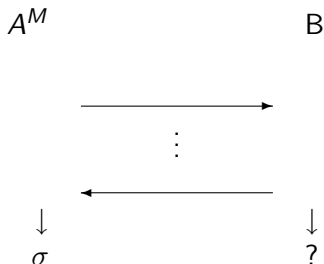


because she does not want Bob to learn M .

Blind Signatures

Bob (B) has secret signing key sk and public verification key pk .

A blind signature scheme provides a signing protocol



At the end of the protocol

- A obtains B 's signature on M .
- But B does not learn M .

Blind Signatures with RSA

Bob (B) has secret signing key N, d and public verification key N, e .
The signature of M is $H(M)^d \bmod N$.

How about

$$\begin{array}{ccc} A^M & & B \\ y \leftarrow H(M) & \xrightarrow{\quad y \quad} & \end{array}$$

Bob gets $H(M)$ but H is one-way (even a RO) so Bob does not learn M .

Blind Signatures with RSA

Bob (B) has secret signing key N, d and public verification key N, e .
The signature of M is $H(M)^d \bmod N$.

How about

$$\begin{array}{ccc} A^M & & B \\ y \leftarrow H(M) & \xrightarrow{\quad y \quad} & \end{array}$$

Bob gets $H(M)$ but H is one-way (even a RO) so Bob does not learn M .

Right?

Blind Signatures with RSA

Bob (B) has secret signing key N, d and public verification key N, e .
The signature of M is $H(M)^d \bmod N$.

How about

$$\begin{array}{ccc} A^M & & B \\ y \leftarrow H(M) & \xrightarrow{\quad y \quad} & \end{array}$$

Bob gets $H(M)$ but H is one-way (even a RO) so Bob does not learn M .

Right?

Of course not. We may know that M comes from a small space S and then Bob can figure out M via

for all $x \in S$ **do**
 if $H(x) = y$ **then return** x

Blind Signatures with RSA

Bob (B) has secret signing key N, d and public verification key N, e .
The signature of M is $H(M)^d \bmod N$.

Signing protocol:

$$\begin{array}{ccc} & A^M & B \\ & R \xleftarrow{\$} \mathbf{Z}_N & \\ y \leftarrow H(M) \cdot R^e \bmod N & \xrightarrow{y} & \\ & \xleftarrow{x} & x \leftarrow y^d \bmod N \\ \sigma \leftarrow x \cdot R^{-1} \bmod N & & \end{array}$$

Modulo N we have

$$x \equiv y^d \equiv H(M)^d \cdot R^{ed} \equiv H(M)^d \cdot R$$

so σ is Bob's signature on M . On the other hand Bob does not learn M .

A really simple protocol

Bob (B) has secret signing key sk and public verification key pk .

$$\begin{array}{ccc} & A^M & B \\ & \longleftarrow sk & \\ \sigma \stackrel{\$}{\leftarrow} \mathcal{S}_{sk}(M) & & \end{array}$$

This satisfies blindness :

- A got B 's signature σ on M
- B did not get M

A really simple protocol

Bob (B) has secret signing key sk and public verification key pk .

$$\begin{array}{ccc} & A^M & B \\ & \longleftarrow sk & \\ \sigma \stackrel{\$}{\leftarrow} \mathcal{S}_{sk}(M) & & \end{array}$$

This satisfies blindness :

- A got B 's signature σ on M
- B did not get M

However it allows A to forge the signature of any message henceforth.

We need to ask for some unforgeability property and check that Chaum's protocol meets it, but what exactly is this property?

The one-more-forgery definition [PS]

- Adversary gets to engage in q interactions with the signer
- Wins if it outputs $q + 1$ valid message signature pairs $(M_1, \sigma_1), \dots, (M_{q+1}, \sigma_{q+1})$.

Unforgeability of Chaum's scheme [BNPS]

The standard one-wayness assumption on RSA does not seem enough to guarantee one-more-unforgeability of Chaum's blind RSA signature scheme.

What has been shown [BNPS]: The scheme is one-more-unforgeable in the RO model under the one-more-RSA-inversion assumption.

Anonymity in payments

Types of payment systems :

- Checks
- Credit Cards
- Cash

Do you want your bank to know where you shop and what you buy?
They will if you use checks or credit cards.

Cash is the only form of payment that provides anonymity: banks, credit agencies etc cannot track and observe your spending patterns.

Types of payment systems

- Electronic checks
- Credit card purchases over Internet
- E-cash?

The first two are not anonymous. Is there some "e-cash" system that will provide anonymity?

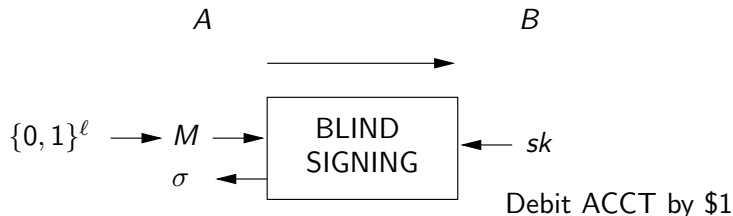
Electronic coins

A \$1 coin in any pair (M, σ) where

- M is the coin serial number
- σ is the bank's signature on M

We will only have \$1 denomination coins in our system

Withdrawal




- Withdrawer Alice picks random serial number M and obtains bank B's signature σ of M to constitute coin (M, σ)
- Bank does not learn M and if it later sees (M, σ) deposited, cannot tell it was withdrawn by Alice.

Alice

Merchant

(M, σ)

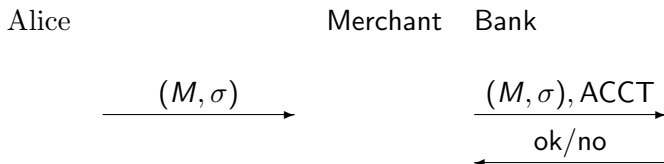


- Alice sends merchant coin (M, σ)
- Merchant verifies that σ is Bank's signature on M

Double spending

The obvious and major problem with e-coins is double spending:

- Alice uses (M, σ) to pay Merchant 1
- Then again uses (M, σ) to pay Merchant 2



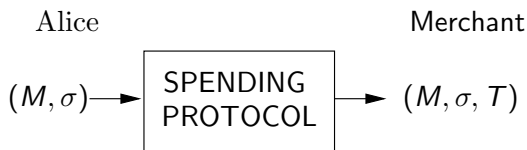
- Bank maintains a database of serial numbers of spent coins
- When merchant gets a coin (M, σ) it at once provides (M, σ) to the bank together with the merchant account number
- If serial number M is not marked as spent then bank deposits \$1 in ACCT, marks M as spent, and says “ok” to the merchant, who now completes transaction with Alice
- If M is marked as spent, bank tells merchant, who aborts transaction with Alice

- Deposits can be made at any time, not necessarily online
- So how is double spending prevented?
- If Alice double-spends, her anonymity is lost and bank can prosecute her
- But as long as she single-spends, she has anonymity

Offline e-cash

Withdrawal: Alice's identity A is encoded in the coin (M, σ) she withdraws.

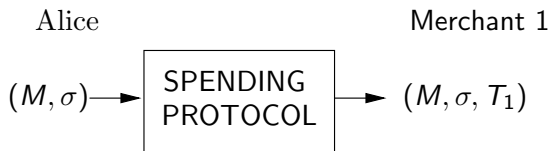
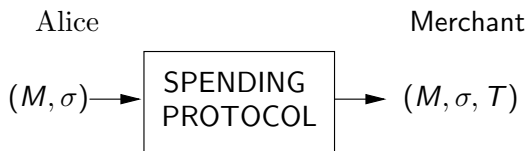
Payment:



T is tracing information

Deposit: Later (offline) merchant can deposit (M, σ, T) at bank, but bank cannot tell the coin came from Alice

Double spending



Now if (M, σ, T_1) and (M, σ, T_2) are deposited then from T_1, T_2 , the bank can obtain Alice's identity

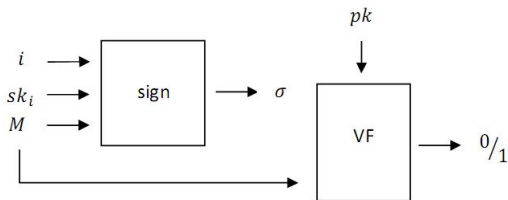
Group signatures [CV]: Anonymity

Group members $1, 2, \dots, n$

Single group public key pk

Member i has secret signing key sk_i

Any group member can create a signature under pk using its secret key



Anonymity: Given (M, σ) one cannot determine the identity $i \in \{1, \dots, n\}$ of the signer

Group signatures: Traceability

- There is a group manager GM who has a special tracing key tk
- The GM can identify the signer from a signature
- Group signatures thus provide revocable anonymity

Many new applications of group signatures are emerging

- Vehicular
- Anonymous routing