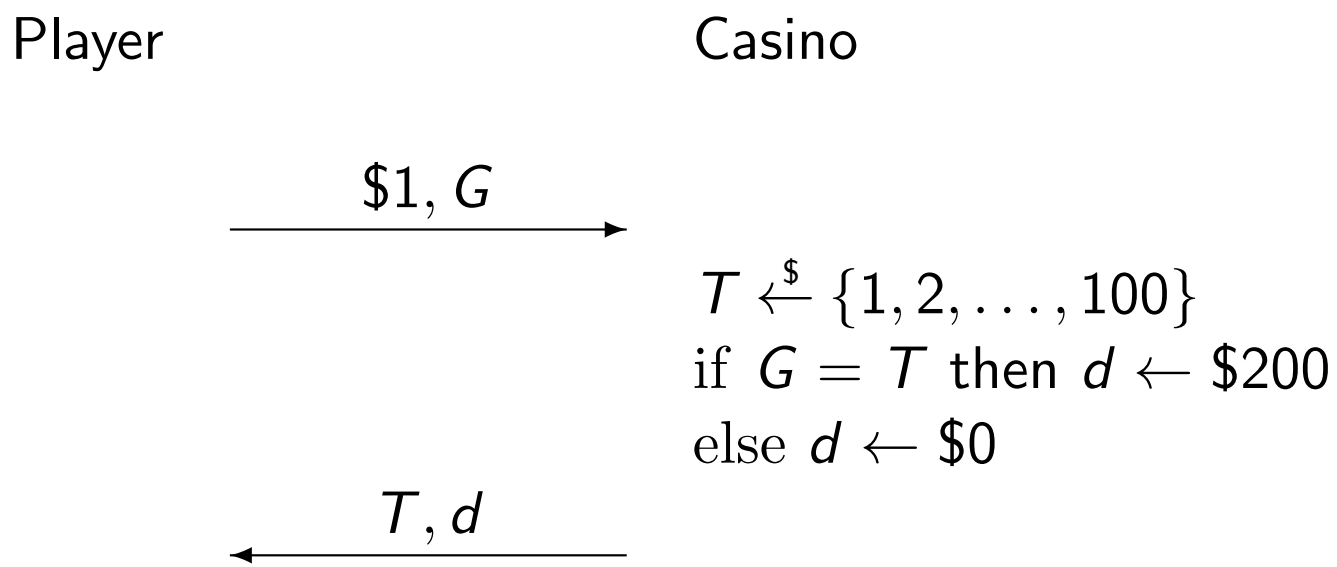
APPLICATIONS AND PROTOCOLS

Some applications and protocols

- Internet Casino
- Commitment
- Shared coin flips
- Threshold cryptography
- Forward security
- Program obfuscation
- Zero-knowledge
- Certified e-mail

- Identity-based encryption
- Fully-homomorphic encryption
- Searchable encryption
- Oblivious transfer
- Secure computation
- Group signatures
- Aggregate signatures
- Electronic voting
- Auctions

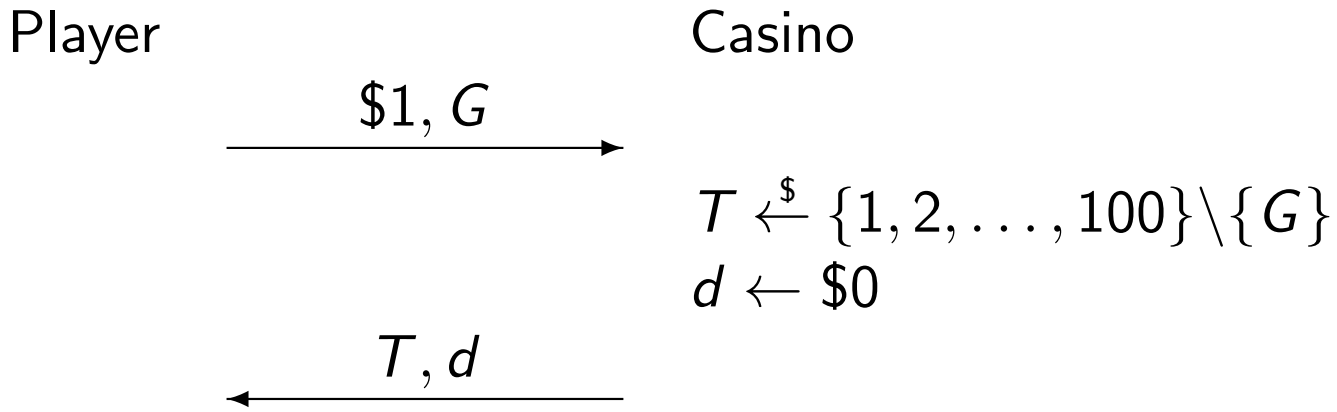
Internet Casino: Protocol G1



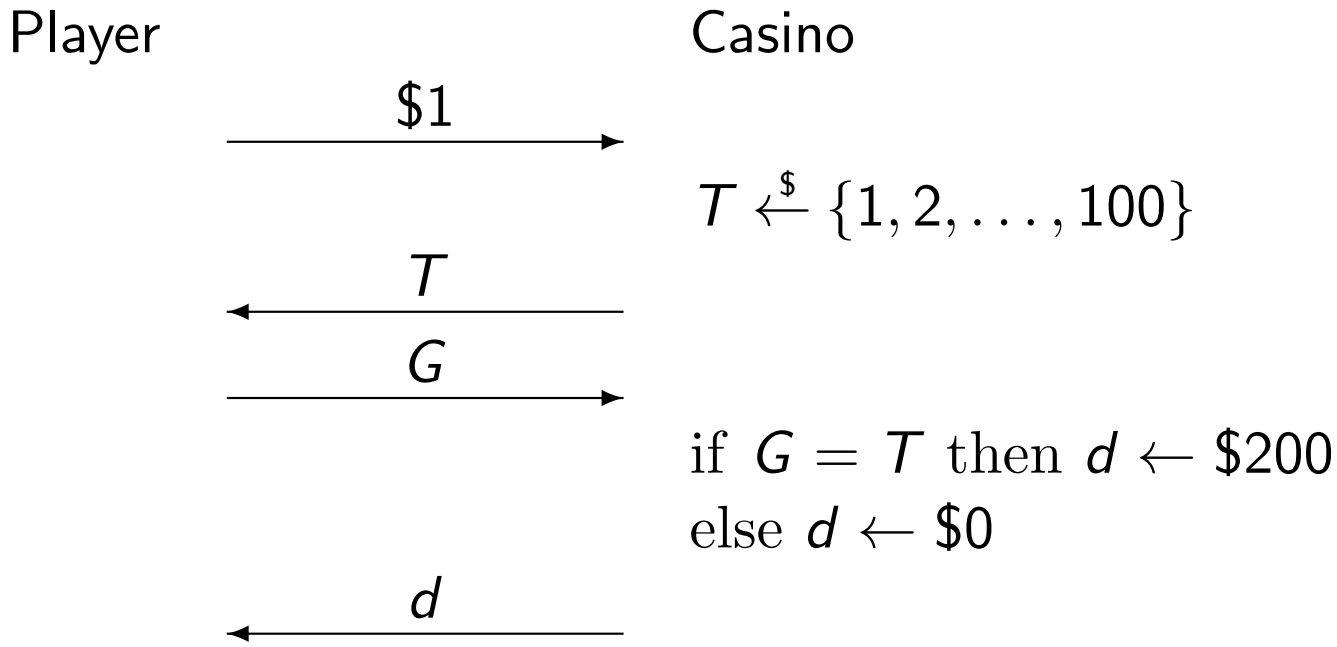
Would you play?

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

Problem: Casino can cheat



Internet Casino: Protocol G2



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

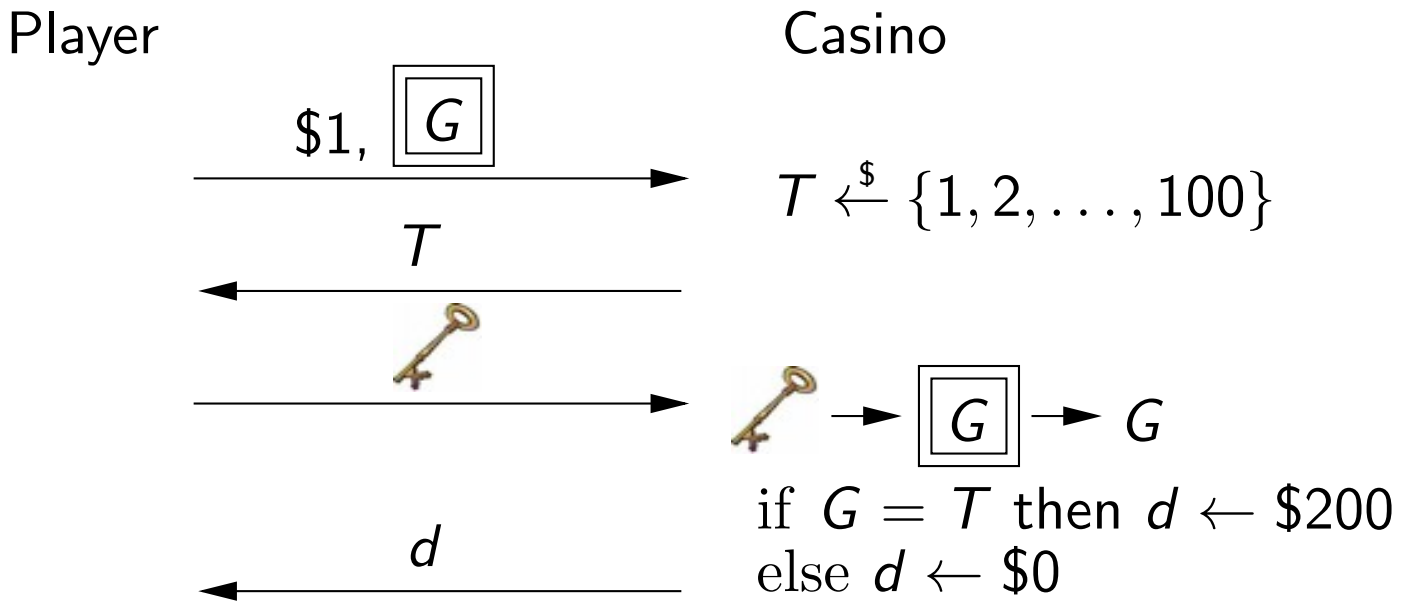
Internet Casino problem

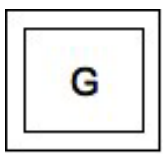
Player and Casino need to exchange G, T so that

- 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

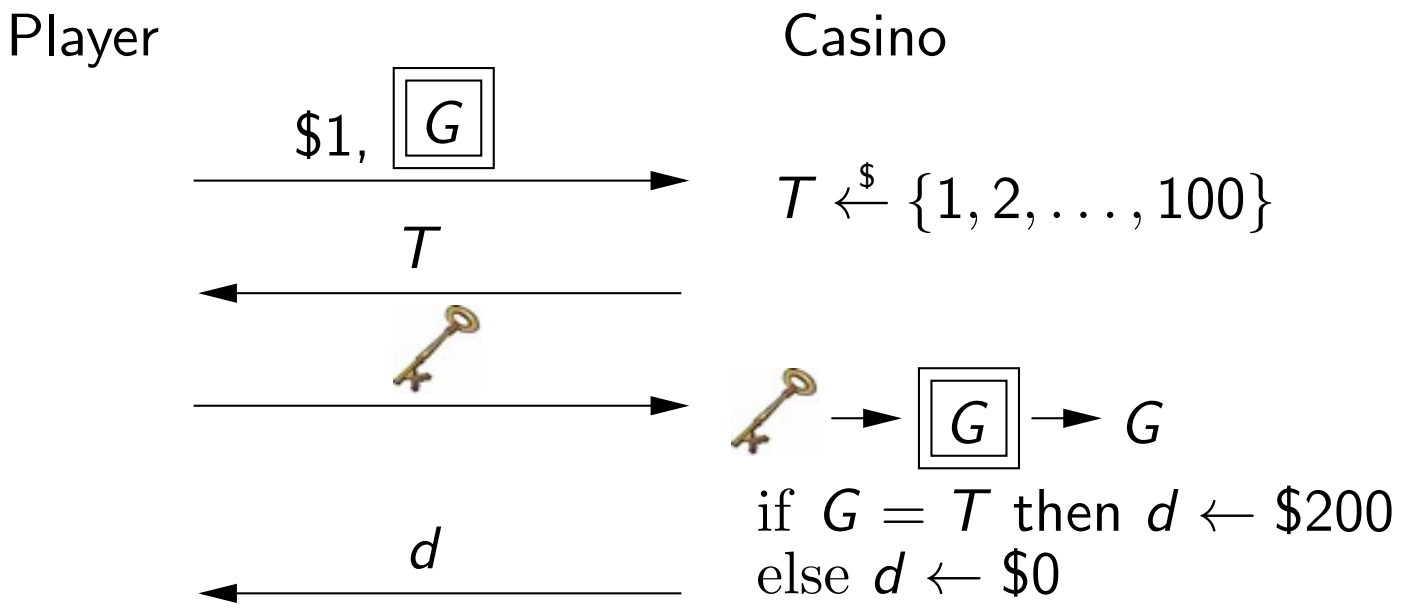




is a locked safe containing a piece of paper with G written on it.

is a key to open the safe.

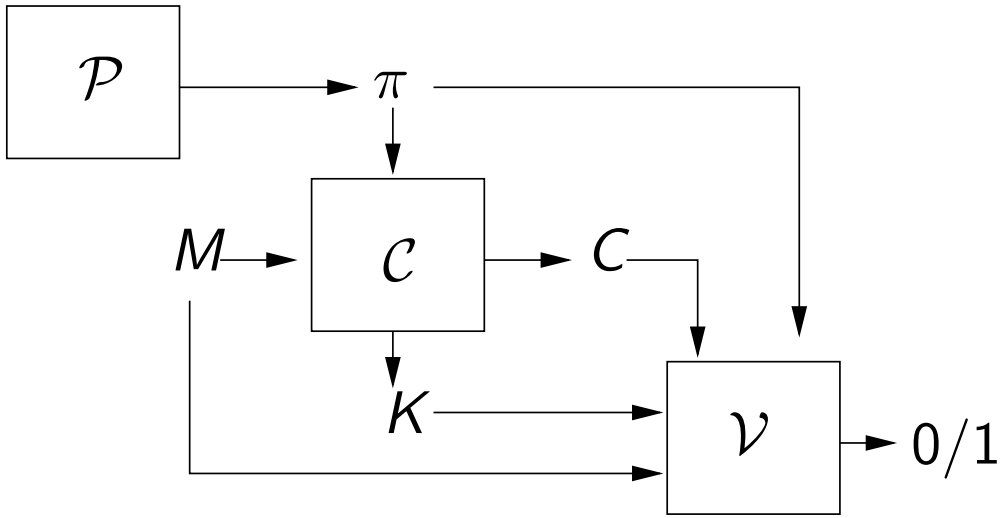
"Internet" Casino: Protocol G3



- 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 P is run once by a trusted party to produce public parameters π.

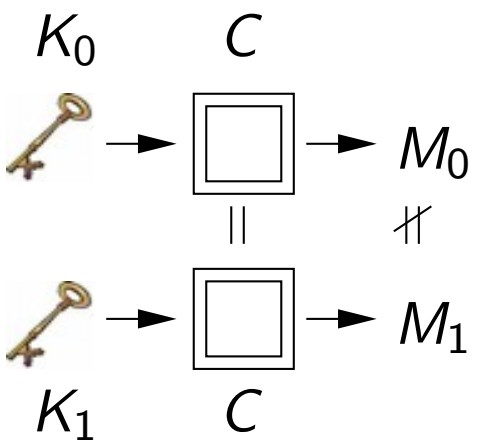
		In Internet Casino
M	Data being commited	G
C	Commital	
K	Decommital key	

Security properties

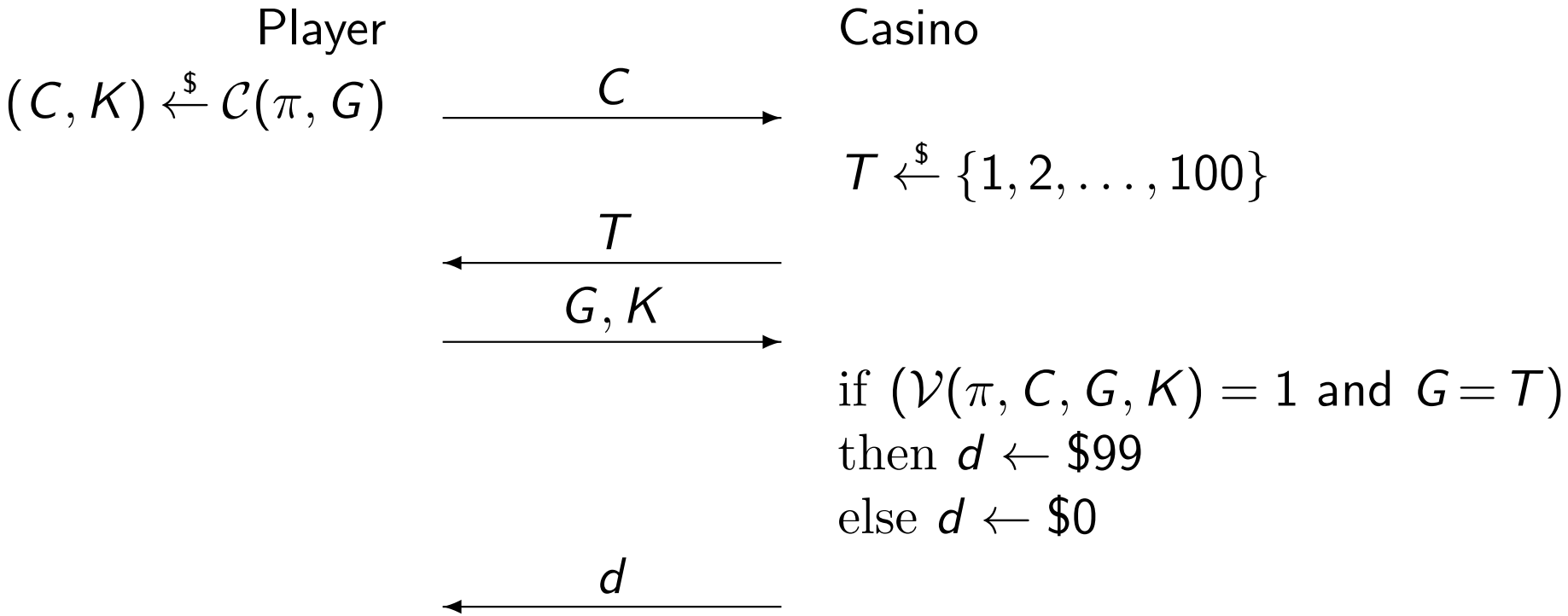
- Hiding: A commital C generated via $(C, K) \xleftarrow{\$} \mathcal{C}(\pi, M)$ should not reveal information about M.

$= C$ does not reveal 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 $\mathcal{V}(\pi, C, M_0, K_0) = \mathcal{V}(\pi, C, M_1, K_1) = 1$.



Internet Casino Protocol using a commitment scheme



Hiding Formally

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

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

procedure Initialize
$\pi \xleftarrow{\$} \mathcal{P}$; $b \xleftarrow{\$} \{0,1\}$
return π

procedure 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

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

Binding Formally

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

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

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

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

Commitment from symmetric encryption

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

$\underline{\textbf{Alg } \mathcal{C}(\pi, M)}$	$\underline{\textbf{Alg } \mathcal{V}(\pi, C, M, K)}$
$K \xleftarrow{\$} \mathcal{K}$; $C \xleftarrow{\$} \mathcal{E}_K(M)$	if $\mathcal{D}_K(C) = M$ then return 1
return (C, K)	else return 0

Is this secure?

Commitment from symmetric encryption

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

$\underline{\textbf{Alg } \mathcal{C}(\pi, M)}$	$\underline{\textbf{Alg } \mathcal{V}(\pi, C, M, K)}$
$K \xleftarrow{\$} \mathcal{K}$; $C \xleftarrow{\$} \mathcal{E}_K(M)$	if $\mathcal{D}_K(C) = M$ then return 1
return (C, K)	else return 0

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 IND-CPA-secure 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.

Commitment from public key encryption

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an IND-CPA-secure 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

$$\begin{array}{l} \underline{\textbf{Alg}\ \ \mathcal{C}(\pi, M)} \\ C \leftarrow H(M);\ K \leftarrow M \\ \text{return } (C, K) \end{array} \quad \Bigg| \quad \begin{array}{l} \underline{\textbf{Alg}\ \ \mathcal{V}(\pi, C, M, K)} \\ \text{return } (C = H(M) \textit{ and } M = K) \end{array}$$

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

$$\begin{array}{l|l} \underline{\textbf{Alg}\ \ \mathcal{C}(\pi, M)} & \underline{\textbf{Alg}\ \ \mathcal{V}(\pi, C, M, K)} \\ C \leftarrow H(M);\ K \leftarrow M & \\ \text{return } (C, K) & \text{return } (C = H(M) \textit{ and } M = K) \end{array}$$

This is

- Binding if H is collision-resistant.
- But not hiding. For example in the Internet Casino $M = G \in \{1, ..., 100\}$ so given $C = H(M)$ the casino can recover M via

$$\begin{array}{l} \text{for } i = 1, ..., 100 \text{ do} \\ \quad \text{if } H(i) = C \text{ then return } i \end{array}$$

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

$$\begin{array}{l|l} \underline{\textbf{Alg}\ \ \mathcal{C}(\pi, M)} & \\ K \xleftarrow{\$} \{0,1\}^{128} & \underline{\textbf{Alg}\ \ \mathcal{V}(\pi, C, M, K)} \\ C \leftarrow H(K||M) & \text{return } (C = H(K||M)) \\ \text{return } (C, K) & \end{array}$$

This is

- Binding if H is collision-resistant (CR).
- One can give an example of CR H such that it is not hiding.
- But for "real" H such as SHA1 it seems hiding in the sense that no attacks are known.

Commitment from hashing

If H is a random oracle then $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$ is a secure hiding and binding commitment scheme where $\mathcal{P}$ returns $\pi = \varepsilon$ and

$$\begin{array}{l|l} \underline{\textbf{Alg}\ \ \mathcal{C}^H(\pi, M)} & \\ K \xleftarrow{\$} \{0,1\}^{128} & \underline{\textbf{Alg}\ \ \mathcal{V}^H(\pi, C, M, K)} \\ C \leftarrow H(K||M) & \text{return } (H(K||M) = C) \\ \text{return } (C, K) & \end{array}$$

There are quite efficient non-RO model commitment schemes based on CR hashing and pairwise-independent hashing [DPW, HM]

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

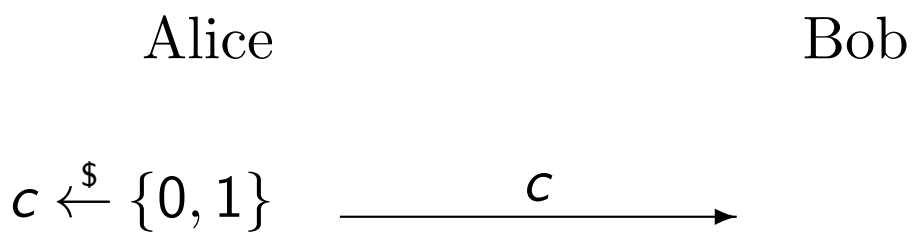
- Alice and Bob are getting divorced
- They want to decide who keeps the Lexus
- They aggree to flip a coin, but
- Alice is in NY and Bob is in LA

Protocol CF1:

Flipping a common coin

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

Protocol CF1:

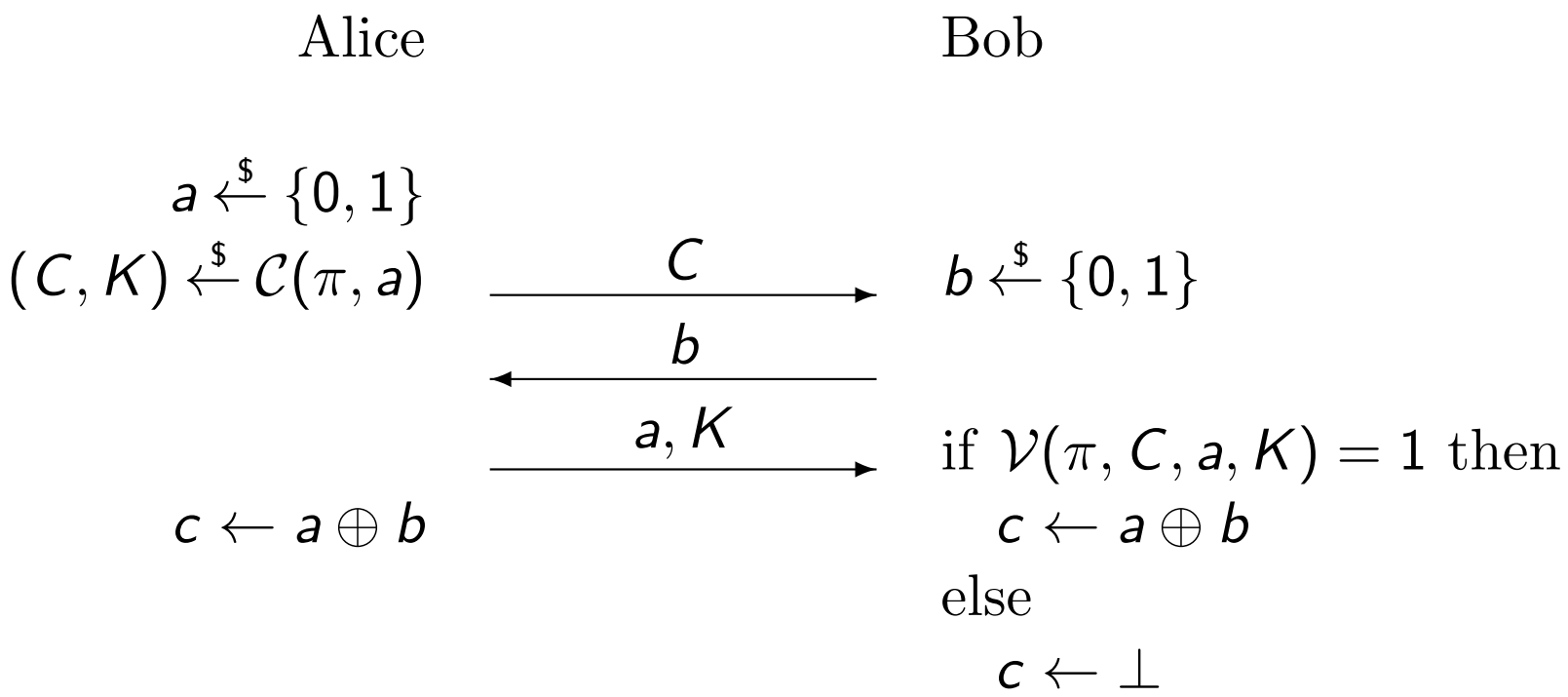


Bob is not too smart but he doesn't like it...

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? Two approaches:

- 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" $d_1, ..., d_n$, 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 $sk = N, d$. Let $P = \varphi(N)$.

Signatures: $\mathcal{S}_{N,d}(M) = H(M)^d \bmod N$.

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

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

How? Let $d_1, d_2, d_3, d_4 \xleftarrow{\$} Z_P$ and let $d_5 \leftarrow d - (d_1 + d_2 + d_3 + d_4) \bmod P$.

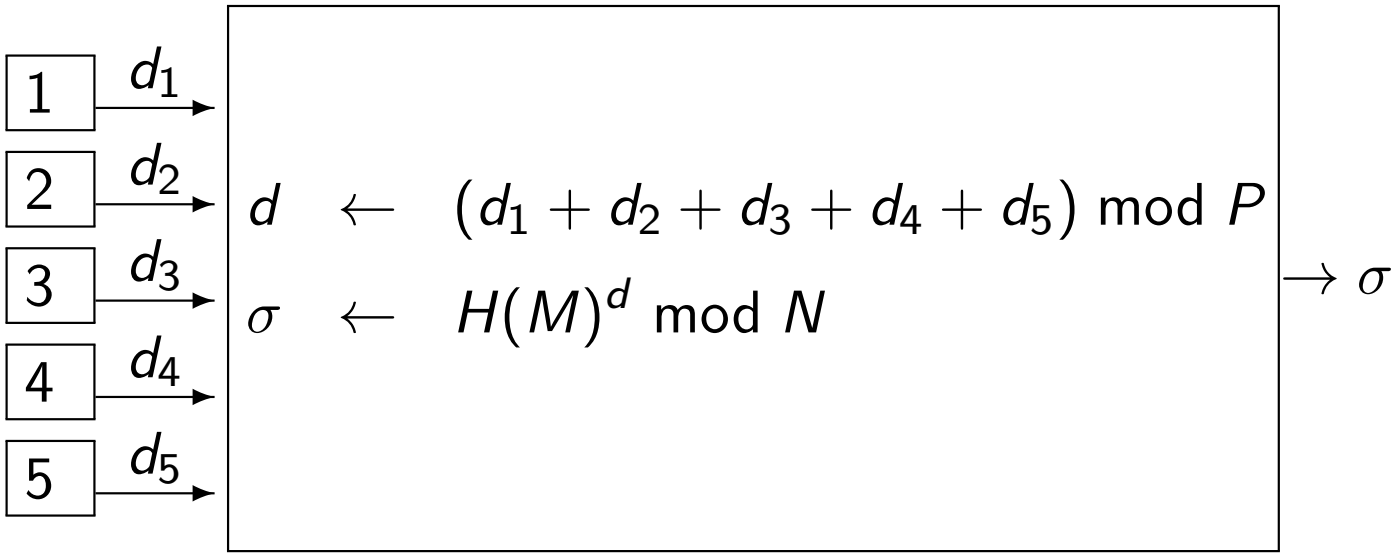
Distribution: Store d_i on server i $(1 \leq i \leq 5)$ and delete d.

Given N, e and any $t = 4$ or less of the 5 shares $d_1, ..., d_5$, it is hard to figure out 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 who generates the signature:



Problems:

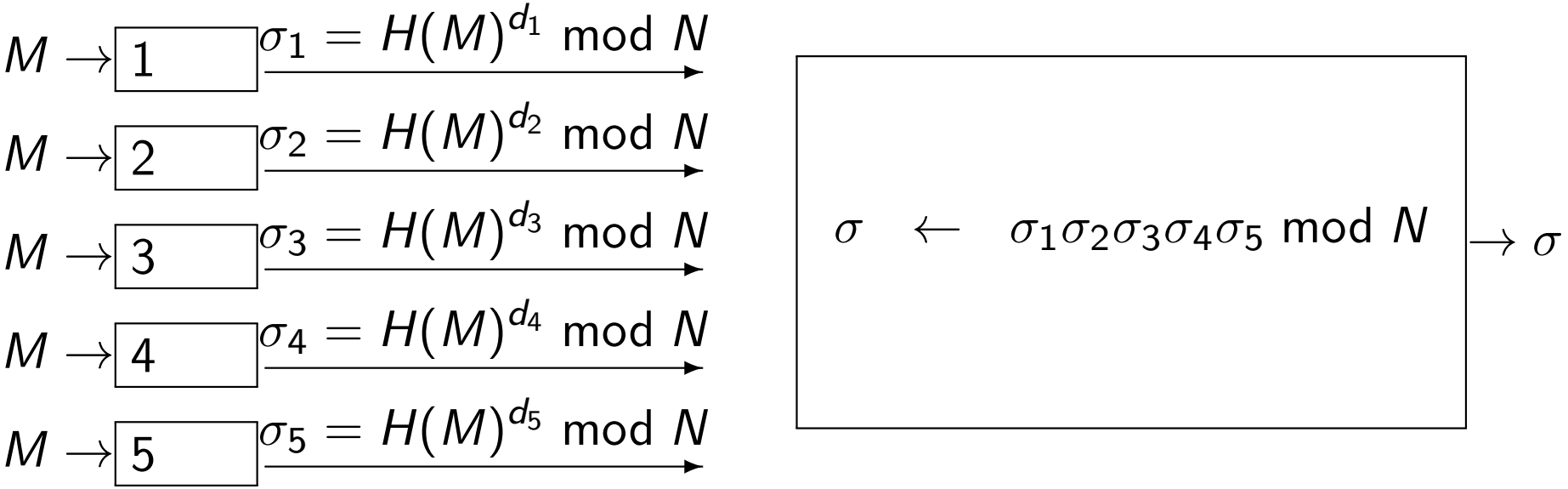
- d is present on Combiner system, even if ephemerally, and can be exposed.
- Combiner stores $P = \varphi(N)$, but exposure of P enables recovery of d.

Threshold signing

If $d = (d_1 + d_2 + d_3 + d_4 + d_5) \bmod P$ then

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

So we could sign M via:



Security of threshold scheme

Let the adversary (1) Specify any set $T \subseteq \{1, ..., 5\}$ of its choice with $|T| \leq 4$, and be given d_i for all $i \in T$ (2) See all communications from servers to Combiner (3) Mount a chosen-message attack to obtain signatures of messages of its choice.

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

Security of threshold scheme

Let the adversary (1) Specify any set $T \subseteq \{1, ..., 5\}$ of its choice with $|T| \leq 4$, and be given d_i for all $i \in T$ (2) See all communications from servers to Combiner (3) Mount a chosen-message attack to obtain signatures of messages of its choice.

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

In particular if H is a RO and RSA is one-way then the threshold scheme is secure.

Security of threshold scheme

Let the adversary (1) Specify any set $T \subseteq \{1, ..., 5\}$ of its choice with $|T| \leq 4$, and be given d_i for all $i \in T$ (2) See all communications from servers to Combiner (3) Mount a chosen-message attack to obtain signatures of messages of its choice.

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

In particular if H is a RO and RSA is one-way then the threshold scheme is secure.

Project: Prove the above: (1) Give a formal syntax and game based definition of security for threshold signature schemes (2) Formally specify the above scheme for general n (3) Formally state and prove the above theorem.

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

January 8: Alice uses her secret signing key sk to produce a signature σ of a document M. Here (M, σ) could be a certificate issued by CA Alice, M could be a contract or M could be a check payable to Bob.

February 1: sk is exposed and Alice revokes her public key pk.

February 11: Bob receives (M, σ). Then Bob will (correctly!) reject it as unauthentic since pk has been revoked.

This is a major nuisance ... if Alice is a CA than all prior certificates she has issued are now void.

Time stamping does not solve this

January 8: Alice uses her secret signing key sk to produce a signature σ on $01/08/2015\|M$.

February 1: sk is exposed and pk is revoked.

February 11: Bob receives $(01/08/2015\|M, \sigma)$ and knows pk was revoked only on 02/01/2015. So should he accept $(01/08/2015\|M, \sigma)$?

Time stamping does not solve this

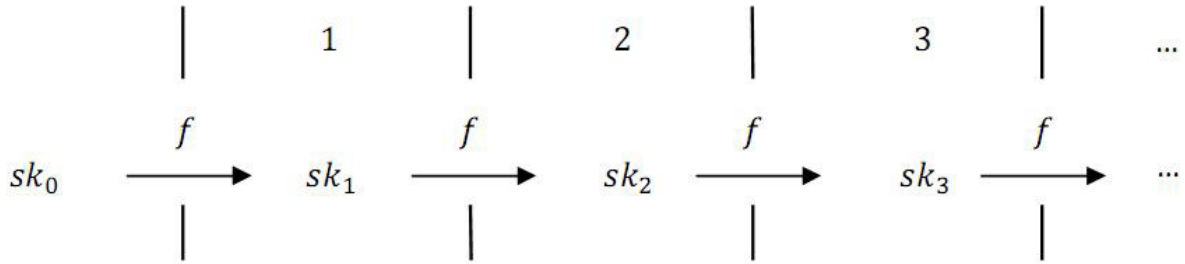
January 8: Alice uses her secret signing key sk to produce a signature σ on $01/08/2015\|M$.

February 1: sk is exposed and pk is revoked.

February 11: Bob receives $(01/08/2015\|M, \sigma)$ and knows pk was revoked only on 02/01/2015. So should he accept $(01/08/2015\|M, \sigma)$?

NO! Adversary obtaining sk on 02/01/2015 could create a signature on $\texttt{mm/dd/yyyy}\|M$ for any $\texttt{mm/dd/yyyy}$ of her choice, even past ones.

Forward security [An, BeMi99]



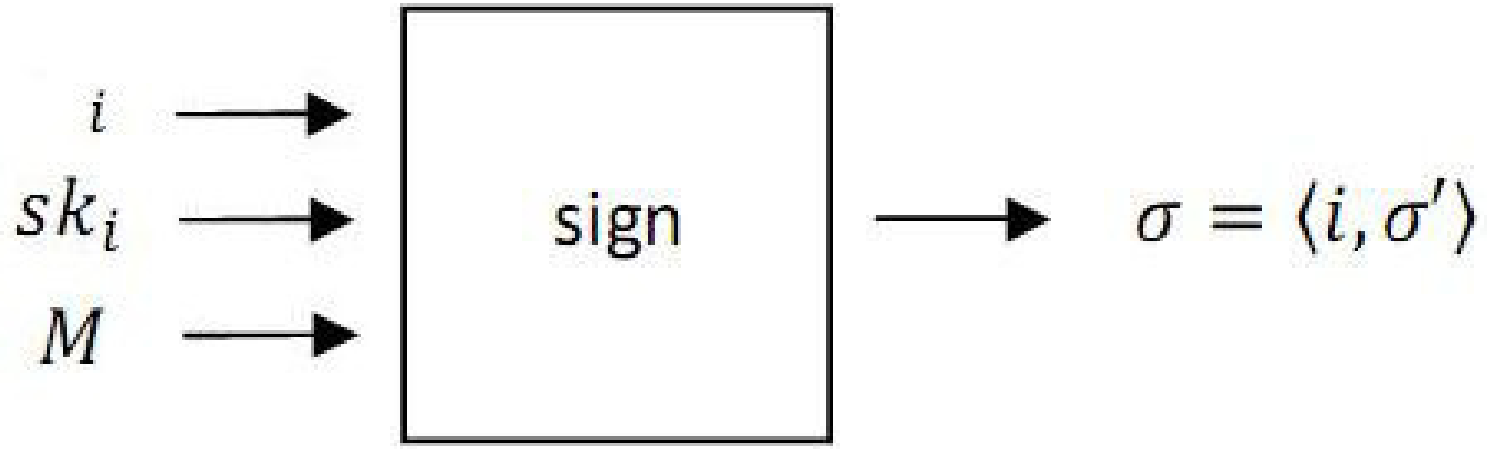
pk : signer's public key, used to verify all the signatures. Certified as usual

sk_i : secret signing key in time period i

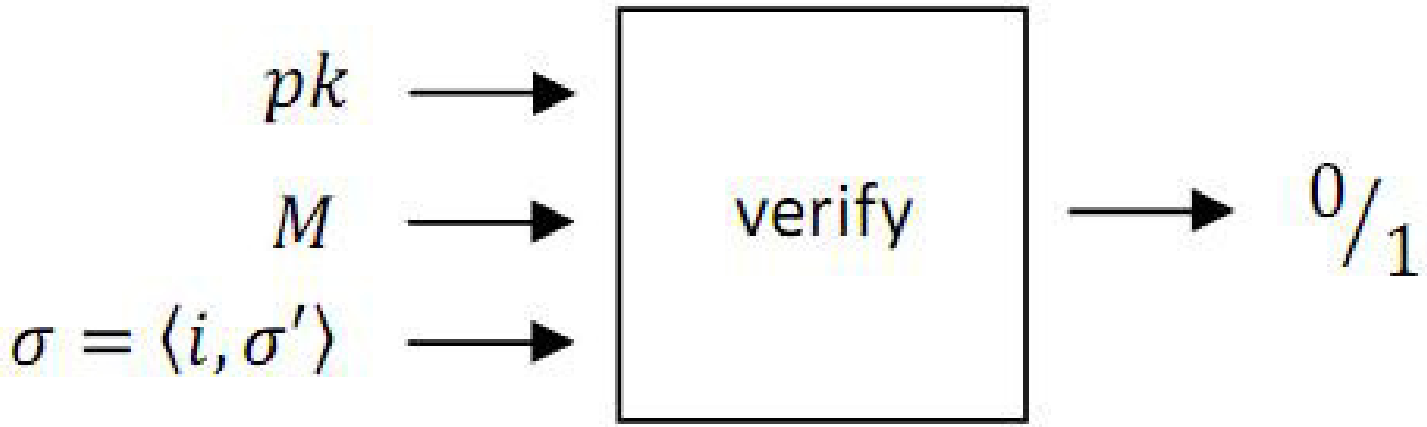
Key evolution: At the start of time period i, the signer

- 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 $sk_{b+1}, sk_{b+2}, \ldots$ 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}, ... exposed and pk revoked for $t = 32$
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.

Schemes and extensions

- 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 $1, ..., n$
Party i has an integer x_i
The parties want to know the value of

$$f(x_1, .., x_n) = x_1 + ... + x_n$$

Secure summation

Suppose we have n parties $1, ..., n$
Party i has an integer x_i
The parties want to know the value of

$$f(x_1, .., x_n) = x_1 + ... + x_n$$

Easy: Let

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

Secure summation

Suppose we have n parties $1, ..., n$
Party i has an integer x_i
The parties want to know the value of

$$f(x_1, .., x_n) = x_1 + ... + x_n$$

Easy: Let

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

Secure summation

Party i has input x_i $(1 \leq i \leq n)$. The parties want to know $f(x_1, .., x_n) = x_1 + ... + x_n$ 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 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 $f(x_1, ..., x_n)$.

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

For example you, as player i, enter x_i into some app on your cellphone which then communicates with the cellphones of the other parties. At the end, the sum shows up on your screen. Take your phone apart and examine all memory contents and you still will not discover x_j for $j \neq i$.

Setup for secure communication protocol

Let N be such that $x_1, ..., x_n \in Z_N = \{0, ..., N-1\}$.
Let $M = nN$.
Let S denote $x_1 + ... + x_n$.
We will compute S mod M, which is just S since

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

Protocol step 1: secret sharing

For $i = 1, ..., n$ party i

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

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

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

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

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$C_1 \quad C_2 \quad C_3 \quad C_4$$

For $j = 1, ..., n$ party j

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

Observation: $S \equiv (C_1 + \cdots + C_n) \pmod{M}$.

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

Security of the protocol

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

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$c_1 \quad c_2 \quad c_3 \quad c_4$$

At end of protocol, party 1 knows (1) x_1, and the first-row entries of the matrix (2) the sum $S = x_1 + x_2 + x_3 + x_4$ (3) c_1, c_2, c_3, c_4 (4) the first column entries $x_{1,1}, x_{2,1}, x_{3,1}, x_{4,1}$.

Claims:

- Party 1 learn nothing about x_4
- Even if parties $1, 2$ pool their information, they learn nothing about x_4
- ...

Secure summation project

Project: Analyze and prove secure the summation protocol: (1) Give a game based definition of privacy (2) Prove that the protocol meets it.

Secure Computation

Parties $1, ..., n$
Party i has private input x_i
They want to compute $f(x_1, ..., x_n)$

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

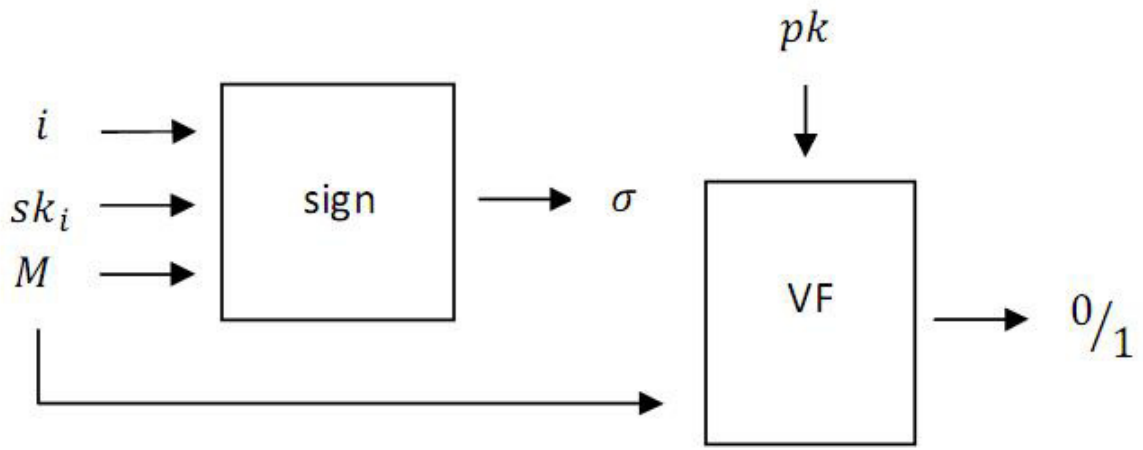
A protocol is t-private if any t parties, getting together, cannot figure out anything about the input of the other parties other than implied by the value of $f(x_1, ..., x_n)$.

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

Enormous body of research.

Group signatures [CV]

Group members $1, 2, ..., n$
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



Group signatures properties [BMW]

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

Traceability:

- There is a group manager GM who has a spacial tracing key tk
- The GM can identify the signer from a signature

Group signatures thus provide revocable anonymity.

Applications of group signatures have been claimed or explored for vehicle communications.