

# KEY EXCHANGE CONTINUED

## Session key exchange requirements

We consider the unilateral, public-key setting. Here  $B$  has a certificate  $\text{CERT}[B]$  and corresponding public and secret keys  $pk[B], sk[B]$ .  $A$  is not assumed to have a certificate or corresponding keys.

This is the most common setting for TLS, where  $B$  is a server like `google.com` and  $A$  is a client.

The session key exchange should result in a session key  $K$ , known to both  $A$  and  $B$ , and satisfying:

- Authenticity:  $A$  really shares  $K$  with  $B$ , not some other entity
- Secrecy: The adversary does not know  $K$ .

This must hold even if the adversary knows session keys of other sessions and is active, meaning in complete control of the communication.

These basic requirements are supplemented by various others including forward secrecy, anonymity, ...

## Session key exchange secrecy

Secrecy: The adversary  $E$  cannot distinguish the true session key  $K$  from a random string of the same length.

Suppose the protocol terminates and a party  $X$  outputs a session key  $K$ . Now we let

$$b \xleftarrow{\$} \{0, 1\}; K_1 \leftarrow K; K_0 \xleftarrow{\$} \{0, 1\}^{|K|}; b' \leftarrow E(K_b)$$

Then the adversary's advantage  $2 \Pr[b = b'] - 1$  should be small.

This must hold even if the adversary has obtained the session key of all other instances except the one partnered with  $X$ , and when the adversary is active, in charge of all communication.

Warning: This is not a formal definition, just a glimpse of it.

# Session key exchange landscape

Session-key exchange is a subtle problem.

Easy to specify protocols, hard to get them right.

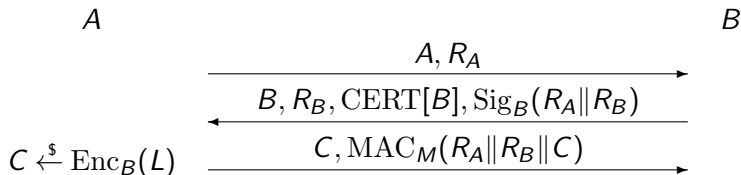
Many security requirements, many proposed protocols, many attacks.

Definitions and provable security treatment started with [BR93] and continued with [BCK98,BPR00,CK01,CK02,...].

Today, standards look for proof-based support.

The TLS 1.3 session key exchange protocol is based on the Sigma protocol of [Kr03].

# Protocol KE1



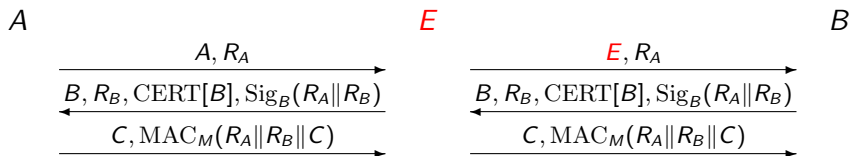
$R_A, R_B$ , called *nonces*, are randomly chosen by the parties.

$\text{Sig}_B(X)$  is  $B$ 's signature on  $X$ , computed under  $sk[B]$  and verifiable under the  $pk[B]$  that is in  $\text{CERT}[B]$ .

$L$  is randomly chosen by  $A$ . Session key is  $K = H_1(L)$  and MAC key is  $M = H_2(L)$  where  $H_1, H_2$  are public hash functions.

$\text{Enc}_B(L)$  is encryption of  $L$  under  $B$ 's public key  $pk[B]$ . Decryption uses  $sk[B]$ .

# Identity mis-binding attack on KE1



A accepts B and thinks it shares  $K$  with B.

But B accepts E and thinks it shares  $K$  with E.

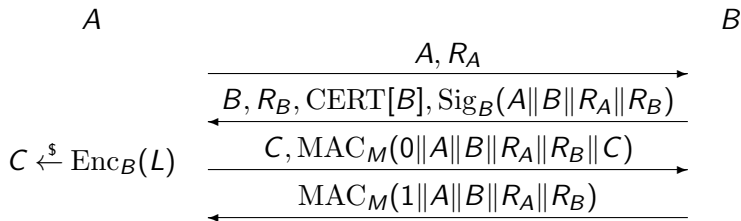
This is viewed as a problem, even though E does not know  $K$ , because there is a mis-binding of identities.

A good definition would view this as a successful attack.

A good protocol should ensure that if A accepts B with  $K$ , then B either accepts A with  $K$ , or accepts nobody with  $K$  or a key related to  $K$ .

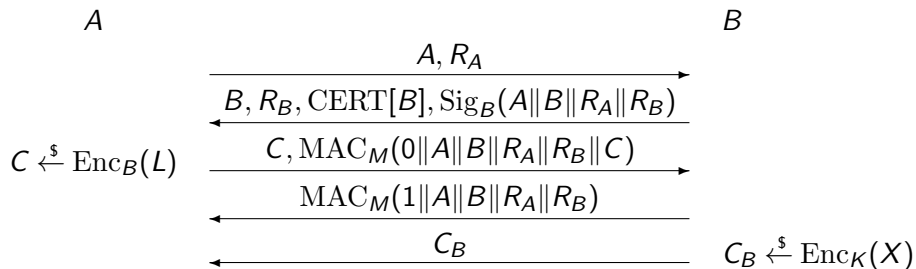
# Protocol KE2

Identity mis-binding is circumvented by inclusion of identities in the signature and the MAC, and addition of a MAC from the server:



Session key is  $K = H_1(A\|B\|R_A\|R_B\|L)$  and MAC key is  $M = H_2(A\|B\|R_A\|R_B\|L)$ .

## KE2 is not forward secure



Nov. 20: Adversary  $E$  records above flows.

Dec. 18:  $E$  compromises  $B$ 's system and obtains  $sk[B]$

Dec. 19:  $B$  revokes  $\text{CERT}[B]$ , and thus  $pk[B]$

However, at any time after Dec. 18,  $E$  can obtain session key  $K$  and decrypt  $C_B$  to obtain  $X$  via:  $K \leftarrow \text{Dec}_{sk[B]}(C)$  ;  $X \leftarrow \text{Dec}_K(C_B)$ .

This is a violation of what's called *forward secrecy*.

*Forward secrecy* asks that exposure of  $sk[B]$  does not allow recovery of session keys  $K$  exchanged prior to the time of exposure.

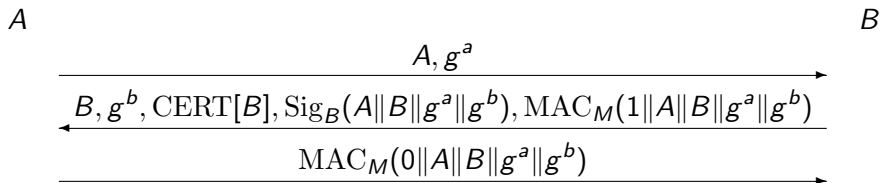
This is achieved using the DH key exchange inside the session key exchange protocol.

Forward secrecy is considered necessary in modern session key exchange, and is present in the TLS 1.3 protocol.

Session-key exchange protocols using DH for forward secrecy are often called authenticated DH key exchange protocols.

# Protocol KE3

Let  $G = \langle g \rangle$  be a cyclic group of order  $m$  in which the CDH problem is hard.

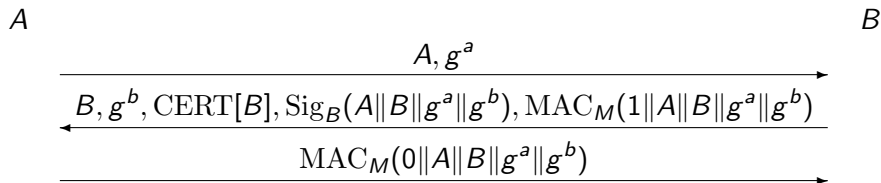


Here  $a, b \xleftarrow{\$} Z_m$  are chosen by  $A, B$ , respectively, and  $g^a, g^b$  play the role of nonces.

$\text{Sig}_B(X)$  is  $B$ 's signature on  $X$ , computed under  $sk[B]$  and verifiable under the  $pk[B]$  that is in  $\text{CERT}[B]$ .

Let  $L = g^{ab}$  be the DH key. Then session key is  $K = H_1(A\|B\|g^a\|g^b\|L)$  and MAC key is  $M = H_2(A\|B\|g^a\|g^b\|L)$  where  $H_1, H_2$  are as before.

# Protocol KE3



There is no public-key encryption used here, only signatures.

Compromise of  $sk[B]$  only gives  $E$  the ability to forge signatures. Even given  $sk[B]$ , it cannot recover the DH key  $L = g^{ab}$  from a prior exchange, and thus cannot distinguish from random the session key  $K = H_1(A\|B\|g^a\|g^b\|L)$ .

Accordingly this provides forward secrecy.

This is roughly the core of the unilateral session-key exchange in the TLS 1.3 handshake. It is based on Sigma [Kr03].

A password is a human-memorizable key.

Attackers can form a set  $D$  of possible passwords called a dictionary such that

- If the target password  $\text{pwd}$  is in  $D$ , and also
- The attacker knows  $\overline{\text{pwd}} = f(\text{pwd})$ , the image of  $\text{pwd}$  under some public function  $f$ ,

then the target password  $\text{pwd}$  can be found via:

For all  $\text{pwd}' \in D$  do

    If  $f(\text{pwd}') = \overline{\text{pwd}}$  then return  $\text{pwd}'$

This is called a dictionary, or brute-force, attack.

# Password usage

Passwords are in widespread use for client authentication to Internet services and servers like gmail, Amazon, Internet banking, ...

Most of us have more passwords than we can remember.

Passwords are communicated over TLS. The main threat is dictionary attacks arising from the adversary obtaining the image  $\overline{\text{pwd}} = f(\text{pwd})$  of the target password  $\text{pwd}$  under some public function  $f$ .

Studies show that many users select poor passwords, meaning ones that fall into attacker dictionaries. And attackers get better and better at making dictionaries. So preventing dictionary attacks is important for security.

# Popular passwords

In 2016, the 25 most common passwords made up more than 10% of surveyed passwords, with the most common making up 4%.

Top 25 most common passwords by year according to SplashData

Rank	2011 <sup>[4]</sup>	2012 <sup>[5]</sup>	2013 <sup>[6]</sup>	2014 <sup>[7]</sup>	2015 <sup>[8]</sup>	2016 <sup>[3]</sup>	2017 <sup>[9]</sup>	2018 <sup>[10]</sup>
1	password	password	123456	123456	123456	123456	123456	123456
2	123456	123456	password	password	password	password	password	password
3	12345678	12345678	12345678	12345	12345678	12345	12345678	123456789
4	qwerty	abc123	qwerty	12345678	qwerty	12345678	qwerty	12345678
5	abc123	qwerty	abc123	qwerty	12345	football	12345	12345
6	monkey	monkey	123456789	123456789	123456789	qwerty	123456789	111111
7	1234567	letmein	111111	1234	football	1234567890	letmein	1234567
8	letmein	dragon	1234567	baseball	1234	1234567	1234567	sunshine
9	trustno1	111111	iloveyou	dragon	1234567	princess	football	qwerty
10	dragon	baseball	adobe123 <sup>[a]</sup>	football	baseball	1234	iloveyou	iloveyou
11	baseball	iloveyou	123123	1234567	welcome	login	admin	princess
12	111111	trustno1	admin	monkey	1234567890	welcome	welcome	admin
13	iloveyou	1234567	1234567890	letmein	abc123	solo	monkey	welcome
14	master	sunshine	letmein	abc123	111111	abc123	login	666666
15	sunshine	master	photoshop <sup>[a]</sup>	111111	1qaz2wsx	admin	abc123	abc123
16	ashley	123123	1234	mustang	dragon	121212	starwars	football
17	bailey	welcome	monkey	access	master	flower	123123	123123
18	passw0rd	shadow	shadow	shadow	monkey	passw0rd	dragon	monkey
19	shadow	ashley	sunshine	master	letmein	dragon	passw0rd	654321
20	123123	football	12345	michael	login	sunshine	master	!@#%*&*
21	654321	jesus	password1	superman	princess	master	hello	charlie
22	superman	michael	princess	696969	qwertyuiop	hottie	freedom	aa123456
23	qazwsx	ninja	azerty	123123	solo	lovere	whatever	donald
24	michael	mustang	trustno1	batman	passw0rd	zaq1zaq1	qazwsx	password1
25	Football	password1	000000	trustno1	starwars	password1	trustno1	qwerty123

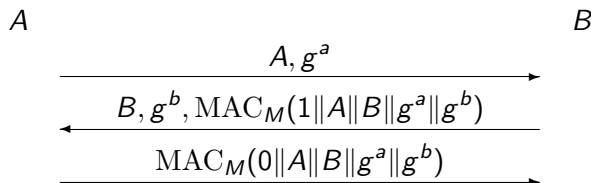
A protocol for Password Authenticated Key Exchange (PAKE) assumes client  $A$  has a password  $\text{pwd}$  and server  $B$  has either  $\text{pwd}$  or its hash under a public hash function.

The parties interact to arrive at a common session key  $K$  satisfying authenticity, secrecy, forward secrecy and also *security against off-line dictionary attacks*.

This means the protocol never reveals an image  $\overline{\text{pwd}} = f(\text{pwd})$  of  $\text{pwd}$  under a public function  $f$ . So even if the password is in the dictionary, the off-line dictionary attack is infeasible.

Roughly, one adversary interaction with one of the parties can eliminate at most one candidate password from the dictionary.

Authentication here is mutual, and no PKI / certificates are assumed.

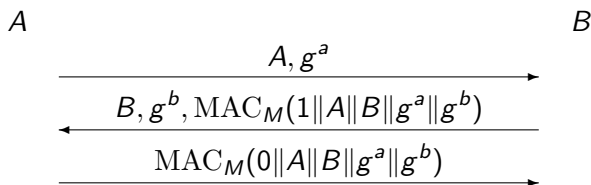


Client  $A$  has password  $\text{pwd}$  that is known to server  $B$ .

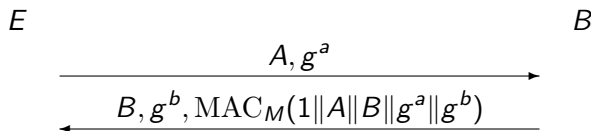
Let  $L = g^{ab}$  be the DH key. Then the session key and MAC keys are  $K = H_1(A\|B\|g^a\|g^b\|L\|\text{pwd})$  and  $M = H_2(A\|B\|g^a\|g^b\|L\|\text{pwd})$ , respectively.

Is this secure against dictionary attack?

# Protocol KE4



A successful dictionary attack by adversary  $E$  is possible, as follows:



$E$  has  $A, B, g^a, g^b$  and also  $L = g^{ab} = (g^b)^a$ . Let

$$f(\text{pwd}) = \text{MAC}_{\text{H}_2(A\|B\|g^a\|g^b\|L\|\text{pwd})}(A\|B\|g^a\|g^b).$$

This  $f$  is a public function of the password, allowing  $E$  to mount the dictionary attack.

# History and status of PAKE

The first protocols were by Bellare and Merritt, 1992.

Definitions and proven-secure protocols begin with [BPR00].

Large literature.

A representative modern PAKE protocol is OPAQUE [JKX18].

# ADVANCED PRIMITIVES AND PROTOCOLS

# Advanced primitives and protocols

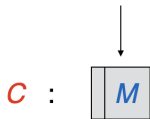
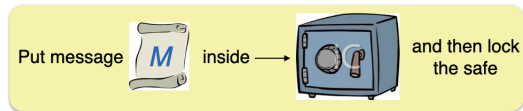
A large body of work on cryptography for goals beyond secure communication.

Usually concerned with privacy in broader settings.

Encompasses computing on encrypted data, secure two- and multi- party computation protocols, zero-knowledge ...

We start with safes and commitment schemes ...

# Safes and their properties



$C$  is a locked safe containing  $M$

Combination safe



Alice knows the secret combination / key **92093**.

key



Unlock  $C$  and remove contents

**Hiding:** Without key  $K$ , one cannot recover the content of locked safe  $C$ .

**Binding:** A single, locked safe  $C$  cannot admit two keys  $K_1, K_2$  that open it to reveal different content  $M_1, M_2$ .

# Commitment schemes

Commitment schemes are, at first cut, a cryptographic (mathematical, digital, ...) way to realize safes.

To be more accurate, a commitment scheme is a cryptographic primitive whose definition formalizes requirements called hiding and binding. A safe is a rough physical analogy, or metaphor, for a commitment scheme.

As with all metaphors, it has its limits, so try to understand commitment schemes via the definitions rather than solely via the metaphor.

Zen saying: The finger pointing at the moon is not the moon ...

Commitment schemes are used in many protocols, including zero-knowledge protocols.

# Syntax of a Commitment Scheme

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

- $\pi \xleftarrow{\$} \mathcal{P}$  — a trusted party runs the parameter generation algorithm  $\mathcal{P}$  to generate public parameters  $\pi$
- $(K, C) \xleftarrow{\$} \mathcal{C}_\pi(M)$  — apply commitment algorithm  $\mathcal{C}$  to message  $M$  to obtain a commitment  $C$  to  $M$  along with a decommitment (or opening) key  $K$ .
- $d \leftarrow \mathcal{V}_\pi(C, M, K)$  — apply verification algorithm  $\mathcal{V}$  to commitment  $C$ , candidate message  $M$  and key  $K$  to obtain a decision  $d \in \{0, 1\}$  as to whether  $C$  is a commitment to  $M$ .

The correctness requirement is that, for all  $\pi$  that may be output by  $\mathcal{P}$ , and all messages  $M$  from the underlying message space, we have  $d = 1$  with probability 1 when  $(K, C) \xleftarrow{\$} \mathcal{C}_\pi(M)$  and  $d \leftarrow \mathcal{V}_\pi(C, M, K)$ .

# Hiding security

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  $\pi$

**procedure** LR( $M_0, M_1$ )

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

return  $C$

**procedure** Finalize( $b'$ )

return  $(b = b')$

The hiding-advantage of  $A$  is

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

Hiding security asks that an adversary having  $C$  but not  $K$  should not learn even partial information about the message  $M$ .

# Binding security

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

Game $\text{BIND}_{\mathcal{CS}}$	<b>procedure</b> Finalize( $C, M_0, M_1, K_0, K_1$ )
<b>procedure</b> Initialize	$v_0 \leftarrow \mathcal{V}_\pi(C, M_0, K_0)$
$\pi \xleftarrow{\$} \mathcal{P}$	$v_1 \leftarrow \mathcal{V}_\pi(C, M_1, K_1)$
return $\pi$	return $((v_0 = 1) \text{ and } (v_1 = 1) \text{ and } (M_0 \neq M_1))$

The binding-advantage of  $A$  is

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

Binding security asks that an adversary be unable to create a commitment  $C$  that it can open to two different messages.

# 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 following commitment scheme:

<u>Alg <math>\mathcal{P}</math></u>	<u>Alg <math>\mathcal{C}_\pi(M)</math></u>	<u>Alg <math>\mathcal{V}_\pi(C, M, K)</math></u>
$\pi \leftarrow \varepsilon$	$K \xleftarrow{\$} \mathcal{K} ; C \xleftarrow{\$} \mathcal{E}_K(M)$	if $\mathcal{D}_K(C) = M$ then return 1
return $\pi$	return $(K, C)$	else return 0

**Q:** Is this hiding?

# 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 following commitment scheme:

<u><b>Alg</b> <math>\mathcal{P}</math></u>	<u><b>Alg</b> <math>\mathcal{C}_\pi(M)</math></u>	<u><b>Alg</b> <math>\mathcal{V}_\pi(C, M, K)</math></u>
$\pi \leftarrow \varepsilon$	$K \xleftarrow{\$} \mathcal{K} ; C \xleftarrow{\$} \mathcal{E}_K(M)$	if $\mathcal{D}_K(C) = M$ then return 1
return $\pi$	return $(K, C)$	else return 0

**Q:** Is this hiding?

YES, since  $\mathcal{SE}$  is IND-CPA.

# 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 following commitment scheme:

<b>Alg</b> $\mathcal{P}$	<b>Alg</b> $\mathcal{C}_\pi(M)$	<b>Alg</b> $\mathcal{V}_\pi(C, M, K)$
$\pi \leftarrow \varepsilon$ return $\pi$	$K \xleftarrow{\$} \mathcal{K} ; C \xleftarrow{\$} \mathcal{E}_K(M)$ return $(K, C)$	if $\mathcal{D}_K(C) = M$ then return 1 else return 0

**Q:** Is this binding?

Not necessarily. For schemes like CTR\$ or CBC\$, the following adversary will have high binding advantage:

**adversary**  $A(\pi)$

$K_0, K_1 \xleftarrow{\$} \{0, 1\}^k ; M_0 \xleftarrow{\$} \{0, 1\}^L ; C \xleftarrow{\$} \mathcal{E}_{K_0}(M_0) ; M_1 \leftarrow \mathcal{D}_{K_1}(C)$   
return  $(C, M_0, M_1, K_0, K_1)$

## 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 following commitment scheme:

<u>Alg <math>\mathcal{P}</math></u>	<u>Alg <math>\mathcal{C}_\pi(M)</math></u>	<u>Alg <math>\mathcal{V}_\pi(C, M, K)</math></u>
$\pi \leftarrow \varepsilon$	$K \xleftarrow{\$} \mathcal{K} ; C \xleftarrow{\$} \mathcal{E}_K(M)$	if $\mathcal{D}_K(C) = M$ then return 1
return $\pi$	return $(K, C)$	else return 0

**Q:** Is this binding if we additionally assume  $\mathcal{SE}$  is INT-CTXT-secure?

The above attack may no longer work. But:

**Exercise:** Show by counter-example that the answer to the above question is NO.

If  $\mathcal{SE}$  is *robust* [ABN10,FLPQ13,FOR17] or *committing* [GLR17] then  $\mathcal{CS}$  will be binding.

# Commitment from hashing

Let  $H: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  be a collision-resistant hash function and let  $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$  be the following commitment scheme:

<u><b>Alg</b> <math>\mathcal{P}</math></u>	<u><b>Alg</b> <math>\mathcal{C}_\pi^H(M)</math></u>	<u><b>Alg</b> <math>\mathcal{V}_\pi^H(C, M, K)</math></u>
$\pi \leftarrow \varepsilon$	$C \leftarrow H(M)$	If $((C = H(M))$ and $(M = K))$
return $\pi$	$K \leftarrow M$	then return 1
	return $(K, C)$	Else return 0

**Q:** Is this binding?

# Commitment from hashing

Let  $H: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  be a collision-resistant hash function and let  $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$  be the following commitment scheme:

<b>Alg</b> $\mathcal{P}$	<b>Alg</b> $\mathcal{C}_\pi^H(M)$	<b>Alg</b> $\mathcal{V}_\pi^H(C, M, K)$
$\pi \leftarrow \varepsilon$	$C \leftarrow H(M)$	If $((C = H(M))$ and $(M = K))$
return $\pi$	$K \leftarrow M$	then return 1
	return $(K, C)$	Else return 0

**Q:** Is this binding?

YES, since  $H$  is collision resistant.

# Commitment from hashing

Let  $H: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  be a collision-resistant hash function and let  $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$  be the following commitment scheme:

<b>Alg</b> $\mathcal{P}$	<b>Alg</b> $\mathcal{C}_\pi^H(M)$	<b>Alg</b> $\mathcal{V}_\pi^H(C, M, K)$
$\pi \leftarrow \varepsilon$	$C \leftarrow H(M)$	If $((C = H(M))$ and $(M = K))$
return $\pi$	$K \leftarrow M$	then return 1
	return $(K, C)$	Else return 0

**Q:** Is this hiding?

# Commitment from hashing

Let  $H: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  be a collision-resistant hash function and let  $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$  be the following commitment scheme:

<b>Alg</b> $\mathcal{P}$	<b>Alg</b> $\mathcal{C}_\pi^H(M)$	<b>Alg</b> $\mathcal{V}_\pi^H(C, M, K)$
$\pi \leftarrow \varepsilon$	$C \leftarrow H(M)$	If $((C = H(M))$ and $(M = K))$
return $\pi$	$K \leftarrow M$	then return 1
	return $(K, C)$	Else return 0

**Q:** Is this hiding?

NO, since  $\mathcal{C}$  is deterministic. Specifically, the following adversary  $A$  has  $\text{Adv}_{\mathcal{CS}}^{\text{hide}}(A) = 1$ :

**adversary**  $A(\pi)$

$C_1 \leftarrow \text{LR}(0, 1)$  ;  $C_2 \leftarrow \text{LR}(1, 1)$

If  $(C_1 = C_2)$  then return 1 else return 0

# Commitment from hashing

Let  $H: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  and let  $CS = (\mathcal{P}, \mathcal{C}, \mathcal{V})$  be the following commitment scheme:

<b>Alg</b> $\mathcal{P}$	<b>Alg</b> $\mathcal{C}_\pi^H(M)$	<b>Alg</b> $\mathcal{V}_\pi^H(C, M, K)$
$\pi \leftarrow \varepsilon$	$K \xleftarrow{\$} \{0, 1\}^\ell$	If $(C = H(K\ M))$
return $\pi$	$C \leftarrow H(K\ M)$	then return 1
	return $(K, C)$	Else return 0

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 SHA256 it seems to be hiding in the sense that no attacks are known.

# Commitment from DL

Let  $G = \langle g \rangle$  be a cyclic group whose order  $m$  is prime. Let  $H: \{0, 1\}^* \rightarrow Z_m$  and let  $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$  be the following commitment scheme:

<b>Alg</b> $\mathcal{P}$	<b>Alg</b> $\mathcal{C}_h^H(M)$	<b>Alg</b> $\mathcal{V}_h^H(C, M, K)$
$x \xleftarrow{\$} Z_m$	$K \xleftarrow{\$} Z_m$	If $(C = g^{H(M)} h^K)$ then return 1
$h \leftarrow g^x$	$C \leftarrow g^{H(M)} h^K$	Else return 0
return $h$	return $(K, C)$	

This is binding if DL is hard in  $G$  and  $H$  is collision-resistant (CR).

This is unconditionally hiding, meaning  $\text{Adv}_{\mathcal{CS}}^{\text{hide}}(A) = 0$  for all  $A$ .

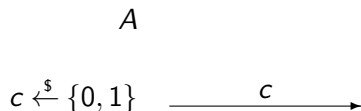
The Pedersen commitment scheme [Pe91] is the special case where the message space is  $Z_m$  and  $H(M) = M$ .

The Pedersen scheme is *homomorphic*: If  $C_1 = g^{M_1} h^{K_1}$  is a commitment to  $M_1$  and  $C_2 = g^{M_2} h^{K_2}$  is a commitment to  $M_2$  then  $C_1 C_2 = g^M h^{K_1+K_2}$  is a commitment to  $M = (M_1 + M_2) \bmod m$ .

# Flipping a common coin

Alice and Bob are getting divorced. They want to flip a common, fair coin  $c$  whose outcome decides which of them keeps the Lexus.

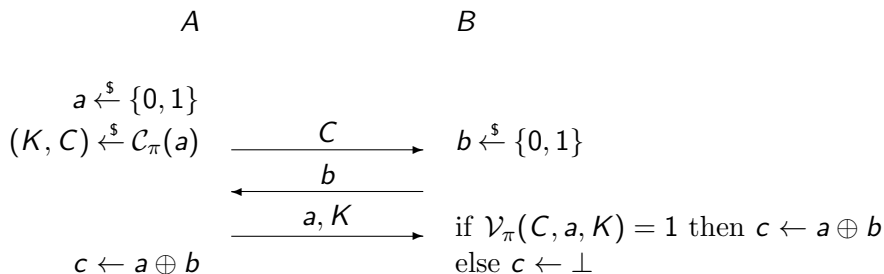
The naive protocol is for  $A$  to flip the coin  $c$  and send it to  $B$ :



But this allows  $A$  to dictate the outcome. Unsurprisingly, she gets the Lexus.

# Flipping a common coin

Let  $\mathcal{CS} = (\mathcal{P}, \mathcal{C}, \mathcal{V})$  be a commitment scheme and consider the following protocol to flip a common coin  $c$ :



The hiding security of  $\mathcal{CS}$  means that  $B$  cannot dictate the outcome  $c$ .

The binding security of  $\mathcal{CS}$  means that  $A$  cannot dictate the outcome  $c$ .

# Homomorphic encryption

Let  $\mathcal{ES} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be an encryption scheme, either symmetric or asymmetric.

We write  $ek$  for the encryption key and  $dk$  for the decryption key. In the symmetric case, they are the same.

We define the *homomorphic evaluation key*  $hk$  to be  $ek$  in the asymmetric case and  $\varepsilon$  in the symmetric case.

Let  $FC$  be a set (class) of functions. We write  $\langle f \rangle$  for a description, for example as a circuit, of a function  $f \in FC$ .

# Homomorphic encryption

$\mathcal{HE}$  is a *homomorphic evaluation algorithm* for  $\mathcal{ES} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  and FC if for all functions  $f \in \text{FC}$  and all messages  $M_1, \dots, M_m$ , where  $m$  is the number of inputs of  $f$ , the following returns true with probability 1:

```
For  $i = 1, \dots, m$  do  $C_i \stackrel{\$}{\leftarrow} \mathcal{E}_{ek}(M_i)$   
 $C \stackrel{\$}{\leftarrow} \mathcal{HE}_{hk}(\langle f \rangle, C_1, \dots, C_m)$  ;  $M \leftarrow \mathcal{D}_{dk}(C)$   
Return ( $M = f(M_1, \dots, M_m)$ )
```

That is,  $C$  is an encryption of  $f(M_1, \dots, M_m)$ .

Encryption scheme  $\mathcal{ES}$  is homomorphic for the class of functions FC if there is an efficient homomorphic evaluation algorithm  $\mathcal{HE}$  as above.

A fully homomorphic encryption (FHE) scheme is one that is homomorphic for the class FC of all functions.

# Homomorphic encryption

**Q:** Isn't homomorphic evaluation always possible, via

**Alg**  $\mathcal{HE}_{hk}(\langle f \rangle, C_1, \dots, C_m)$

For  $i = 1, \dots, m$  do  $M_i \leftarrow \mathcal{D}_{dk}(C_i)$

$M \leftarrow f(M_1, \dots, M_m)$  ;  $C \stackrel{s}{\leftarrow} \mathcal{E}_{ek}(M)$  ; Return  $C$

**A:**  $\mathcal{HE}$  is not given  $dk$ . And the requirement that  $\mathcal{HE}$  is efficient means that it is infeasible for it to compute  $dk$  from  $hk$ .

# Security of homomorphic encryption

The primary security requirement for a homomorphic encryption scheme  $\mathcal{ES}$  is simply IND-CPA.

Sometimes one wants the scheme to be function hiding (FH), which means that, on seeing  $C \stackrel{\$}{\leftarrow} \mathcal{HE}_{hk}(\langle f \rangle, \mathcal{E}_{ek}(M_1), \dots, \mathcal{E}_{ek}(M_m))$ , one does not learn  $f$ . A game-based definition follows.

Sometimes one wants that homomorphically evaluated ciphertexts are distributed just like real ones, meaning the following are indistinguishable:

- $C \stackrel{\$}{\leftarrow} \mathcal{HE}_{hk}(\langle f \rangle, \mathcal{E}_{ek}(M_1), \dots, \mathcal{E}_{ek}(M_m))$
- $C' \stackrel{\$}{\leftarrow} \mathcal{E}_{ek}(f(M_1, \dots, M_m))$ .

## Extended key generation

Let  $\mathcal{ES} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be an encryption scheme, either symmetric or asymmetric. We define its *extended key-generation algorithm*  $\overline{\mathcal{K}}$  via:

If  $\mathcal{ES}$  is symmetric:

**Alg**  $\overline{\mathcal{K}}$

$K \xleftarrow{\$} \mathcal{K}$

$ek \leftarrow K ; dk \leftarrow K ; hk \leftarrow \varepsilon$

Return  $(ek, dk, hk)$

If  $\mathcal{ES}$  is asymmetric:

**Alg**  $\overline{\mathcal{K}}$

$(ek, dk) \xleftarrow{\$} \mathcal{K}$

$hk \leftarrow ek$

Return  $(ek, dk, hk)$

This yields a unified syntax for symmetric and asymmetric schemes.

## Function hiding security

Let  $\mathcal{ES} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be an encryption scheme with homomorphic evaluation algorithm  $\mathcal{HE}$  for the class of functions FC. Let  $A$  be an adversary.

Game  $\text{FH}_{\mathcal{ES}, \mathcal{HE}}$

**procedure** Initialize

$b \xleftarrow{\$} \{0, 1\}$  ;  $i \leftarrow 0$  ;  $(ek, dk, hk) \xleftarrow{\$} \overline{\mathcal{K}}$  ; Return  $hk$

**procedure** Enc( $M$ )

$i \leftarrow i + 1$  ;  $M_i \leftarrow M$  ;  $C_i \xleftarrow{\$} \mathcal{E}_{ek}(M)$  ; Return  $C_i$

**procedure** LR( $i_1, \dots, i_m, f_0, f_1$ )

$C \xleftarrow{\$} \mathcal{HE}_{hk}(\langle f_b \rangle, C_{i_1}, \dots, C_{i_m})$  ; Return  $C$

**procedure** Finalize( $b'$ )

return  $(b = b')$

## Function hiding security

In game  $\text{FH}_{\mathcal{ES}, \mathcal{HE}}$ , any **LR** query  $i_1, \dots, i_m, f_0, f_1$  must satisfy the following conditions:

- $f_0, f_1 \in \text{FC}$
- $m$  is the number of inputs of both  $f_0$  and  $f_1$
- $1 \leq i_1, \dots, i_m \leq i$
- $|f_0(M_{i_1}, \dots, M_{i_m})| = |f_1(M_{i_1}, \dots, M_{i_m})|$ .

The fh-advantage of  $A$  is

$$\text{Adv}_{\mathcal{ES}, \mathcal{HE}}^{\text{fh}}(A) = 2 \cdot \Pr \left[ \text{FH}_{\mathcal{ES}, \mathcal{HE}}^A \Rightarrow \text{true} \right] - 1 .$$

We (informally) say that  $(\mathcal{ES}, \mathcal{HE})$  is FH-secure for FC if, as usual, any practical adversary  $A$  has low fh-advantage.

# Homomorphic encryption can't be IND-CCA

If an encryption scheme  $\mathcal{ES} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  is homomorphic for a non-trivial class of functions  $\text{FC}$  then it cannot be IND-CCA-secure.

Why? Assume the adversary  $A$  can find  $M_0, M_1$  and  $f \in \text{FC}$  such that

$$(1) \quad f(M_0) \neq M_0 \text{ and } f(M_1) \neq M_1$$

$$(2) \quad f(M_0) \neq f(M_1)$$

Then it can achieve  $\text{Adv}_{\mathcal{ES}}^{\text{ind-cca}}(A) = 1$  via:

**adversary**  $A(hk)$  //  $hk = ek$  (asymmetric) or  $hk = \varepsilon$  (symmetric)

$C \xleftarrow{\$} \text{LR}(M_0, M_1)$  ;  $C' \xleftarrow{\$} \mathcal{HE}_{hk}(\langle f \rangle, C)$  ;  $M' \leftarrow \text{Dec}(C')$

If  $(M' = f(M_1))$  then return 1 else return 0

Condition (1) ensures  $C' \neq C$  so the Dec-query is valid. Then (2) ensures that  $A$ 's output is correct.

# Possible usage of homomorphic encryption

Homomorphic encryption allows computing on encrypted data.

$A$  picks keys  $ek, dk, hk$ , encrypts her data  $M_1, \dots, M_m$  under  $ek$  to get  $C_1, \dots, C_m$ .

$A$  uploads the ciphertexts and  $hk$  to in-the-cloud server  $B$ .

Later  $A$  can send  $\langle f \rangle$  to  $B$ , who computes and returns  $C \stackrel{\$}{\leftarrow} \mathcal{HE}_{hk}(\langle f \rangle, C_1, \dots, C_m)$ .

$A$  now recovers  $M = f(M_1, \dots, M_m) \leftarrow \mathcal{D}_{dk}(C)$ .

# CTR\$ is XOR-homomorphic

Let  $F: \{0,1\}^k \times \{0,1\}^\ell \rightarrow \{0,1\}^L$  be a family of functions and let

$$G_K(R, n) = F_K(R+1) \parallel \cdots \parallel F_K(R+n/L).$$

Then recall the CTR\$ symmetric encryption scheme  $\mathcal{ES} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  has

Alg  $\mathcal{E}_K(M)$

$R \xleftarrow{\$} \{0,1\}^\ell ; n \leftarrow |M|$

$X \leftarrow G_K(R, n) \oplus M$

Return  $R \parallel X$

Alg  $\mathcal{D}_K(R \parallel X)$

$n \leftarrow |X|$

$M \leftarrow G_K(R, n) \oplus X$

Return  $M$

Say  $R \parallel X \xleftarrow{\$} \mathcal{E}_K(M)$  and  $n = |M|$ . If  $\Delta \in \{0,1\}^n$  and  $Y \leftarrow X \oplus \Delta$  then

$$\mathcal{D}_K(R \parallel Y) = G_K(R, n) \oplus Y = G_K(R, n) \oplus X \oplus \Delta = M \oplus \Delta.$$

## CTR\$ is XOR-homomorphic

Let  $B = \{0, 1\}^L$  and let  $B^+$  be the set of all strings whose length is a positive multiple of  $L$ .

For  $\Delta \in B^+$ , define  $f_\Delta: \{0, 1\}^{|\Delta|} \rightarrow \{0, 1\}^{|\Delta|}$  by  $f_\Delta(M) = \Delta \oplus M$ . Let FC be the set of all functions  $f_\Delta$  as  $\Delta$  ranges over  $B^+$ . Let  $\langle f_\Delta \rangle = \Delta$ .

Let  $\mathcal{ES}$  be the CTR\$ symmetric encryption scheme based on family of functions  $F: \{0, 1\}^k \times \{0, 1\}^\ell \rightarrow B$ , as above.

Then the above shows that  $\mathcal{ES}$  is homomorphic for FC, with homomorphic evaluation algorithm:

**Alg**  $\mathcal{HE}_\varepsilon(\Delta, C)$  //  $|C| = \ell + \Delta$   
 $R \parallel X \leftarrow C$  //  $R = \ell$  and  $X \in B^+$   
 $Y \leftarrow X \oplus \Delta$ ; Return  $R \parallel Y$

# An asymmetric XOR-homomorphic scheme

Let  $G = \langle g \rangle$  be a cyclic group of order  $m$ . Let  $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$ . Define  $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  via

$$\begin{array}{l|l|l} \mathbf{Alg} \mathcal{K} & \mathbf{Alg} \mathcal{E}_X^H(M) \quad // |M| = n & \mathbf{Alg} \mathcal{D}_x^H((Y, W)) \\ \hline x \xleftarrow{\$} Z_m & y \xleftarrow{\$} Z_m ; Y \leftarrow g^y ; Z \leftarrow X^y & Z \leftarrow Y^x \\ X \leftarrow g^x & W \leftarrow H(Y\|Z) \oplus M & M \leftarrow H(Y\|Z) \oplus W \\ \text{return } (X, x) & \text{return } (Y, W) & \text{return } M \end{array}$$

Above,  $X$  is the encryption key and  $x$  is the decryption key.

This scheme is IND-CPA in the random oracle model if CDH is hard in  $G$ .

Say  $(Y, W) \xleftarrow{\$} \mathcal{E}_X(M)$ . If  $\Delta \in \{0, 1\}^n$  and  $V \leftarrow W \oplus \Delta$  then

$$\mathcal{D}_x((Y, V)) = H(Y\|Y^x) \oplus V = H(Y\|Y^x) \oplus W \oplus \Delta = M \oplus \Delta.$$

# An asymmetric XOR-homomorphic scheme

For  $\Delta \in \{0, 1\}^n$ , define  $f_\Delta: \{0, 1\}^n \rightarrow \{0, 1\}^n$  by  $f_\Delta(M) = \Delta \oplus M$ . Let FC be the set of all functions  $f_\Delta$  as  $\Delta$  ranges over  $\{0, 1\}^n$ . Let  $\langle f_\Delta \rangle = \Delta$ .

Let  $\mathcal{ES}$  be the above asymmetric encryption scheme.

Then the above shows that  $\mathcal{ES}$  is homomorphic for FC, with homomorphic evaluation algorithm:

**Alg**  $\mathcal{HE}_X^H(\Delta, (Y, W))$  //  $|W| = n$

$V \leftarrow W \oplus \Delta$  ; Return  $(Y, V)$

# Security of the above XOR homomorphic schemes

Both our symmetric and our asymmetric XOR homomorphic encryption schemes, above, are IND-CPA secure.

However they are not FH-secure.

**Exercise:** Specify, in pseudocode, efficient adversaries, for each of the above schemes, that achieve high fh-advantage.

## Further homomorphic encryption schemes

There are many schemes that are homomorphic for operations like addition and multiplication over various groups.

The first FHE (Fully Homomorphic Encryption) scheme was given by Gentry [Ge09]. Many further and simpler ones have been proposed.

A representative example is [\[GSW13\]](#).

Encryption in these schemes is based on matrices. The schemes allow homomorphic evaluation of the addition and multiplication operations on one-bit messages, which implies homomorphic evaluation of all functions.

Security (IND-CPA) is based on the Learning with Errors (LWE) assumption.

These blog posts provide an overview of the ideas: [Part 1](#), [Part 2](#).

There are [many libraries](#) implementing FHE schemes.

# The millionaire's problem

$A$ 's net financial worth is  $x_1$  and  $B$ 's net financial worth is  $x_2$ .

They want to know who is richer, but they don't want to reveal their net financial worth to each other.

Let  $y \leftarrow (x_1 > x_2)$ .

A secure computation protocol allows them to interact in such a way that at the end:

- $A$  gets  $y$  but learns nothing more about  $x_2$  than given by  $y$
- $B$  gets  $y$  but learns nothing more about  $x_1$  than given by  $y$

Note some privacy is lost:

- If  $y = \text{true}$  then  $A$  learns that  $x_2 \in \{0, \dots, x_1 - 1\}$
- else she learns that  $x_2 \in \{x_1 + 1, \dots, \infty\}$

However, she should not learn anything more about  $x_2$  than the above.

## 2-party functionalities

A 2-party functionality  $2\text{-}\mathcal{PF} = (\mathcal{I}, \mathcal{F})$  is a pair of algorithms that operate as follows:

- $(l_1, l_2) \stackrel{\$}{\leftarrow} \mathcal{I}$  — The initialization algorithm  $\mathcal{I}$  generates keys for parties 1, 2 respectively.
- $(y_1, y_2) \leftarrow \mathcal{F}_{(l_1, l_2)}(x_1, x_2)$  — the functionality  $\mathcal{F}$  takes the inputs  $x_1, x_2$  of parties 1, 2, respectively, and (deterministically) returns outputs  $y_1, y_2$  for the parties, respectively.

We denote  $y_i$  by  $\mathcal{F}_{(l_1, l_2)}(x_1, x_2)[i]$ .

We say that  $2\text{-}\mathcal{PF}$  is keyless if  $l_1 = l_2 = \varepsilon$ , in which case we may omit writing  $l_1, l_2$ .

## 2-party functionalities

A 2-party functionality  $2\text{-}\mathcal{PF} = (\mathcal{I}, \mathcal{F})$  is a pair of algorithms that operate as follows:

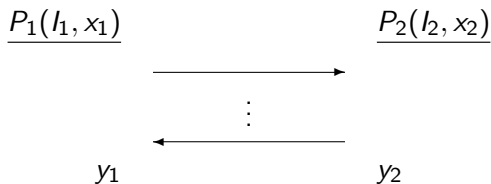
- $(l_1, l_2) \xleftarrow{\$} \mathcal{I}$  — The initialization algorithm  $\mathcal{I}$  generates keys for parties 1, 2 respectively.
- $(y_1, y_2) \leftarrow \mathcal{F}_{(l_1, l_2)}(x_1, x_2)$  — the functionality  $\mathcal{F}$  takes the inputs  $x_1, x_2$  of parties 1, 2, respectively, and (deterministically) returns outputs  $y_1, y_2$  for the parties, respectively.

**Example:** The millionaires problem is captured by the following keyless functionality:

Alg  $\mathcal{F}(x_1, x_2)$

$y \leftarrow (x_1 > x_2)$  ; Return  $(y, y)$

## Secure-computation of a 2-party functionality



Party  $P_i$  is initialized with  $l_i$  and has an input  $x_i$ .

At the end of the protocol,  $P_i$  has an output  $y_i$ .

The syntax of a protocol  $\Pi$  says how parties compute the messages they send each other, and how they determine their outputs.

Correctness asks that  $y_i = \mathcal{F}_{(l_1, l_2)}(x_1, x_2)[i]$ .

Privacy asks that  $P_i$  learns nothing about  $x_{3-i}$  other than what is revealed by the fact that  $y_i = \mathcal{F}_{(l_1, l_2)}(x_1, x_2)[i]$ .

Defining privacy, and even correctness, is delicate. The literature has many settings and definitions.

In the semi-honest, also called honest-but-curious, setting, the parties are assumed to follow the protocol  $\Pi$ .

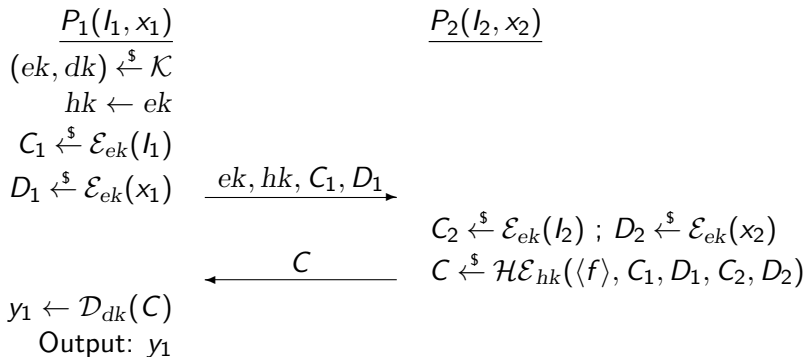
In the malicious setting, a party may deviate from  $\Pi$  in an attempt to violate privacy.

We will only consider the semi-honest setting.

# General Protocol from FHE

Let  $2\text{-}\mathcal{PF} = (\mathcal{I}, \mathcal{F})$  be a (any) 2-party functionality . Assume for simplicity there is no output for  $P_2$ , meaning  $\mathcal{F}_{(l_1, l_2)}(x_1, x_2)[2] = \varepsilon$ .

Let  $\mathcal{ES} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be an asymmetric FHE scheme with homomorphic evaluation algorithm  $\mathcal{HE}$ . Define  $f(l_1, x_1, l_2, x_2) = \mathcal{F}_{(l_1, l_2)}(x_1, x_2)[1]$ . Then the following protocol  $\Pi^{\text{fhe}}$  securely computes  $2\text{-}\mathcal{PF}$ :



Let  $2\text{-}\mathcal{PF} = (\mathcal{I}, \mathcal{F})$  be a (any) 2-party functionality. Then one can give a secure computation protocol  $\Pi^{\text{gc}+\text{ot}}$  for it that uses circuit garbling schemes [BHR12,BHR13] and oblivious transfer (OT).

Garbling schemes and OT are quite fast and getting faster, but  $\Pi^{\text{gc}+\text{ot}}$  is still not practical in many situations. With current FHE schemes,  $\Pi^{\text{fhe}}$  is even less practical.

Thus we seek dedicated (direct) secure computation protocols for 2-party functionalities of practical interest.

We will discuss OPRFs (Oblivious PRFs) and PSM (Private Set Membership).

## PRFs with public keys

Let  $F: \text{Keys} \times D \rightarrow R$  be a function family. A public-key deriver for  $F$  is a function  $P: \text{Keys} \rightarrow \{0, 1\}^*$ . We extend PRF security via:

Game  $\text{Real}_{F,P}$

**procedure** Initialize

$K \xleftarrow{\$} \text{Keys} ; X \leftarrow P(K)$

Return  $X$

**procedure**  $\text{Fn}(x)$

Return  $F_K(x)$

Game  $\text{Rand}_{F,P}$

**procedure** Initialize

$K \xleftarrow{\$} \text{Keys} ; X \leftarrow P(K)$

Return  $X$

**procedure**  $\text{Fn}(x)$

if  $T[x] = \perp$  then  $T[x] \xleftarrow{\$} R$

Return  $T[x]$

The **advantage** of  $A$  is

$$\text{Adv}_{F,P}^{\text{prf}}(A) = \Pr \left[ \text{Real}_{F,P}^A \Rightarrow 1 \right] - \Pr \left[ \text{Rand}_{F,P}^A \Rightarrow 1 \right] .$$

Thus the requirement is that PRF security is maintained even if the adversary has  $X = P(K)$ . We refer to this as PRF security of  $F$  given  $P$ . The prior notion of PRF security corresponds to  $P(K) = \varepsilon$ .

# A CDH-based PRF with a public key

Let  $G = \langle g \rangle$  be a cyclic group of order  $m$ . Let  $H_1: \{0, 1\}^* \rightarrow G$  and  $H_2: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ . Define  $F: Z_m \times \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  and  $P: Z_m \rightarrow G$  via

$$\begin{array}{l|l} \mathbf{Alg} \ F_K^{H_1, H_2}(M) & \mathbf{Alg} \ P(K) \\ \hline Y \leftarrow H_1(M)^K ; z \leftarrow H_2(Y \| M) & X \leftarrow g^K \\ \text{Return } z & \text{Return } X \end{array}$$

Then  $F$  is PRF-secure given  $P$ , in the random oracle model, assuming CDH is hard in  $G$ .

# Oblivious PRFs

Let  $F: \text{Keys} \times D \rightarrow R$  be a function family, and  $P: \text{Keys} \rightarrow \{0,1\}^*$ . In applications, we want  $F$  to be PRF-secure given  $P$ .

An oblivious PRF (OPRF) is captured by the following 2-party functionality  $2\text{-}\mathcal{PF}^{\text{opr}} = (\mathcal{I}, \mathcal{F})$ :

$$\begin{array}{l|l} \mathbf{Alg} \mathcal{I} & \mathbf{Alg} \mathcal{F}_{(X,K)}(M, \varepsilon) \\ \hline K \xleftarrow{\$} \text{Keys} ; X \leftarrow P(K) ; \text{Return } (X, K) & z \leftarrow F_K(M) ; \text{Return } (z, \varepsilon) \end{array}$$

Server  $P_2$  has a key  $K$  for  $F$ . Client  $P_1$  has an input  $M$  for  $F$ .

A secure computation protocol  $\Pi$  for  $2\text{-}\mathcal{PF}^{\text{opr}}$  provides  $P_1$  with  $F_K(M)$ , but the server does not learn  $M$  and the client learns nothing more than  $F_K(M)$ . In particular the client does not learn  $K$ .

# A naive OPRF protocol

Let  $G = \langle g \rangle$  be a cyclic group of order  $m$ . Let  $H_1: \{0, 1\}^* \rightarrow G$  and  $H_2: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ . Let  $F: \mathbb{Z}_m \times \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  be the CDH-based PRF given  $P: \mathbb{Z}_m \rightarrow G$  that was defined above.

$$\begin{array}{ccc} \frac{P_1(X, M)}{B \leftarrow H_1(M)} & \xrightarrow{B} & \frac{P_2(K, \varepsilon)}{Y \leftarrow B^K} \\ & \xleftarrow{Y} & \\ z \leftarrow H_2(Y \| M) & & \\ \text{Output: } z & & \end{array}$$

Correctness holds, meaning  $z = F_K^{H_1, H_2}(M)$ .

But  $P_1$ 's privacy is violated because  $B$  may reveal information about  $M$ . Indeed if  $M$  is drawn from a small, known space, then  $P_2$  can determine  $M$ .

# An OPRF protocol

Let  $G = \langle g \rangle$  be a cyclic group of order  $m$ . Let  $H_1: \{0, 1\}^* \rightarrow G$  and  $H_2: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ . Let  $F: \mathbb{Z}_m \times \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  be the CDH-based PRF given  $P: \mathbb{Z}_m \rightarrow G$  that was defined above.

$$\begin{array}{ccc} \frac{P_1(X, M)}{b \xleftarrow{\$} \mathbb{Z}_m ; B \leftarrow H_1(M)g^b} & \xrightarrow{B} & \frac{P_2(K, \varepsilon)}{Z \leftarrow B^K} \\ & \xleftarrow{Z} & \\ Y \leftarrow ZX^{-b} ; z \leftarrow H_2(Y \| M) & & \\ \text{Output: } z & & \end{array}$$

We refer to  $g^b$  as the blinding factor and to  $B$  as the blinded version of  $H_1(M)$ .

**Correctness:** We have

$$Y = ZX^{-b} = B^K X^{-b} = (H_1(M)g^b)^K g^{-Kb} = H_1(M)^K$$

so  $z = F_K^{H_1, H_2}(M)$ .

# An OPRF protocol

$$\begin{array}{ccc} & \frac{P_1(X, M)}{b \stackrel{s}{\leftarrow} Z_m ; B \leftarrow H_1(M)g^b} & \frac{P_2(K, \varepsilon)}{Z \leftarrow B^K} \\ & \xrightarrow{B} & \\ & \xleftarrow{Z} & \\ Y \leftarrow ZX^{-b} ; z \leftarrow H_2(Y \| M) & & \\ \text{Output: } z & & \end{array}$$

**Privacy for  $P_1$ :** The choice  $b \stackrel{s}{\leftarrow} Z_m$  makes  $g^b$ , and hence  $B$ , a group element that is random and independent of  $M$ . So  $P_2$  learns nothing about  $M$ .

**Privacy for  $P_2$ :** This corresponds to  $F$  being a PRF given  $P$ .

mycallisto.org

Callisto is a serial **sexual assault prevention and protection tool**, supported by a community of survivors.

Victim can identify / report a perpetrator. Perpetrator identity is encrypted and stored by Callisto. If two victims report the same perpetrator (called a match), Callisto detects this, and action is taken.

An OPRF is part of Callisto's protocol.

# Private Set Membership

The private set membership (PSM) 2-party functionality is the keyless functionality  $2\text{-}\mathcal{PF}^{\text{psm}} = (\mathcal{I}, \mathcal{F})$  in which:

**Alg**  $\mathcal{F}(M, S)$

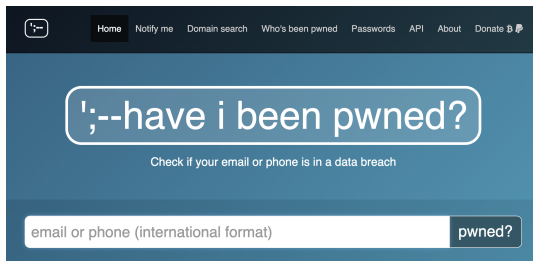
$d \leftarrow (M \in S)$  ; Return  $((d, |S|), \varepsilon)$

Server  $P_2$  has a set  $S$ . Client  $P_1$  has an input  $M$ .

A secure computation protocol  $\Pi$  for  $2\text{-}\mathcal{PF}^{\text{psm}}$  allows  $P_1$  to learn whether or not  $M \in S$ . It also learns the size of  $S$ , but nothing about  $S$  beyond that. The server does not learn  $M$ .

In private set intersection (PSI), the input of  $P_1$  is also a set,  $R$ , and its output is  $R \cap S$ . PSM is the special case in which  $|R| = 1$ .

# Possible uses of PSM: Have I been pwned?



Server  $P_2$  has a set  $S$  of credentials that have been compromised, meaning lost in a data breach. A credential could be a username, password, phone number, ...

Client  $P_2$  wants to know if its credential  $M$  is in  $S$ , meaning has been compromised. Obviously  $P_1$  does not want to reveal its credential to  $P_2$ . Meanwhile, making the set  $S$  public is undesirable. PSM provides a solution.



## Technology preview: Private contact discovery for Signal

moxie0 on 26 Sep 2017

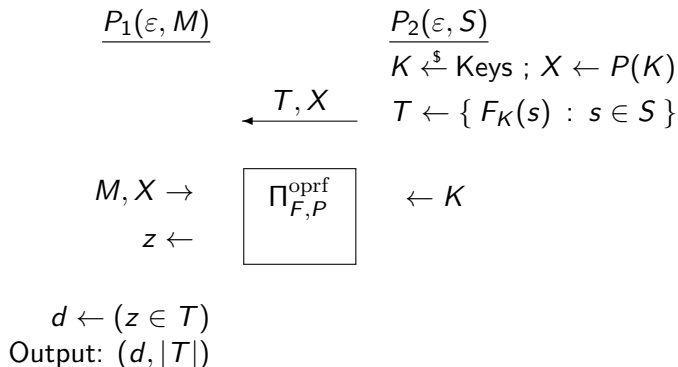
At Signal, we've been thinking about [the difficulty of private contact discovery](#) for a long time. We've been working on strategies to improve our current design, and today we've [published a new private contact discovery service](#).

Using this service, Signal clients will be able to efficiently and scalably determine whether the contacts in their address book are Signal users **without revealing the contacts in their address book to the Signal service**.

PSM can be used by Signal users to determine whether a contact in their address book is a Signal user without revealing their contacts to Signal.

# A PSM protocol

Let  $F: \text{Keys} \times \{0, 1\}^* \rightarrow \mathbb{R}$  be a PRF with public-key deriver  $P$ . Let  $\Pi_{F,P}^{\text{oprfl}}$  be an OPRF protocol for  $F, P$ .



**Correctness:** Assuming  $F_K$  is collision resistant we will have  $d = (M \in S)$  and  $|T| = |S|$ .

# A PSM protocol

$$\underline{P_1(\varepsilon, M)}$$
$$\underline{P_2(\varepsilon, S)}$$
$$K \xleftarrow{s} \text{Keys} ; X \leftarrow P(K)$$
$$T \leftarrow \{ F_K(s) : s \in S \}$$
$$\longleftarrow T, X$$
$$M, X \rightarrow$$
$$z \leftarrow$$
$$\boxed{\Pi_{F,P}^{\text{opr}}}$$
$$\leftarrow K$$
$$d \leftarrow (z \in T)$$

Output:  $(d, |T|)$

**Privacy for  $P_1$ :** The privacy of the OPRF protocol guarantees that  $P_2$  learns nothing about  $M$ .

**Privacy for  $P_2$ :** The PRF-security of  $F$  guarantees that  $P_1$  learns nothing about  $S$  beyond  $(d, |S|)$ .

Let  $1 \leq t < n$ . A  $(t, n)$ -secret-sharing scheme allows an entity, called the dealer, to split  $s$  into shares  $s_1, \dots, s_n$  such that:

- Given any  $t + 1$  shares, one can recover  $s$ , but
- Given any  $t$  or less shares, one learns nothing about  $s$ .

Secret sharing is useful in its own right and also a tool in secure multi-party computation.

# Secret sharing syntax

Let  $1 \leq t < n$ . A  $(t, n)$ -secret-sharing scheme  $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$  is a pair of algorithms that operate as follows:

- $\{(i, s_i)\}_{i=1}^n \stackrel{\$}{\leftarrow} \mathcal{SH}(s)$  — the dealer runs the sharing algorithm  $\mathcal{SH}$  on input a secret  $s \in \{0, 1\}^*$  to get a list of  $n$  shares.
- $s \leftarrow \mathcal{RE}(R, \{(i, s_i)\}_{i \in R})$  — apply deterministic recovery algorithm  $\mathcal{RE}$  to a set  $R \subseteq \{1, \dots, n\}$ , and a list of shares for players in  $R$ , to obtain output  $s \in \{0, 1\}^* \cup \{\perp\}$ .

The correctness requirement is that, for all  $s$  in the underlying message space and all  $R \subseteq \{1, \dots, n\}$  with  $|R| \geq t + 1$  we have:

If  $\{(i, s_i)\}_{i=1}^n \stackrel{\$}{\leftarrow} \mathcal{SH}(s)$  and  $s' \leftarrow \mathcal{RE}(R, \{(i, s_i)\}_{i \in R})$  then  $s' = s$ .

## Secret sharing privacy

Let  $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$  be a  $(t, n)$ -secret-sharing scheme, and  $A$  an adversary.

Game  $\text{IND}_{\mathcal{SS}}$

**procedure** Initialize

$b \xleftarrow{\$} \{0, 1\}$

**procedure** LR( $T, s_0, s_1$ ) //  $|s_0| = |s_1|$  and  $T \subseteq \{1, \dots, n\}$  with  $|T| = t$   
 $\{(i, s_i)\}_{i=1}^n \xleftarrow{\$} \mathcal{SH}(s_b)$  ; return  $\{(i, s_i)\}_{i \in T}$

**procedure** Finalize( $b'$ )

return  $(b = b')$

The ind-advantage of  $A$  is

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

IND-privacy for a secret-sharing scheme asks that knowledge of up to  $t$  (out of  $n$ ) shares, of a sharing of a secret, not yield any partial information about the secret.

We refer to  $T$  as the set of corrupted players. It must have size at most  $t$ .

We say that secret-sharing scheme  $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$  is *perfect*, or has perfect privacy, if  $\text{Adv}_{\mathcal{SS}}^{\text{ind}}(A) = 0$  for all  $A$ .

## A $(n - 1, n)$ perfect secret-sharing scheme

Let  $G$  be a commutative group whose group operation we denote by “+”, the inverse operation being “-”. Examples are:

- $G = \mathbb{Z}_M$  with + being addition modulo  $M$ .
- $G = \{0, 1\}^m$  with + being bitwise XOR.

The shares  $\{(i, s_i)\}_{i=1}^n$  of a secret  $s \in G$  are chosen so that  $s_1, \dots, s_n$  are random elements of  $G$  subject to the condition that

$$s = s_1 + \dots + s_n .$$

**Recovery:** Given  $s_1, \dots, s_n$  we can recover  $s$  via the above equation.

**Privacy:** Any  $n - 1$  of  $s_1, \dots, s_n$  are randomly and independently distributed over  $G$ , so we have perfect privacy.

## A $(n - 1, n)$ perfect secret-sharing scheme

Let  $G$  be a commutative group whose group operation we denote by “+”, the inverse operation being “-”.

We define secret-sharing scheme  $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$  via

<b>Alg</b> $\mathcal{SH}(s)$ // $s \in G$	<b>Alg</b> $\mathcal{RE}(R, \{(i, s_i)\}_{i \in R})$
For $i = 1, \dots, n - 1$ do $s_i \xleftarrow{\$} G$	$s \leftarrow 0$ // Identity element of the group
$s_n \leftarrow s - (s_1 + \dots + s_{n-1})$	For $i \in R$ do $s \leftarrow s + s_i$
return $\{(i, s_i)\}_{i=1}^n$	return $s$

This is a  $(n - 1, n)$  perfect secret-sharing scheme.

For correctness, we are only concerned with  $R = \{1, \dots, n\}$ .

Perfect privacy is because any  $n - 1$  of  $s_1, \dots, s_n$  are randomly and independently distributed over  $G$ .

# Shamir's $(t, n)$ perfect secret-sharing scheme

Let  $F$  be a finite field of size at least  $n + 1$ . Examples are:

- $F = \mathbb{Z}_p$  where  $p \geq n + 1$  is a prime.
- $F = \text{GF}(2^k)$  where  $2^k \geq n + 1$ .

Fix some distinct points  $e_1, \dots, e_n \in F$ .

To share secret  $s \in F$ , Shamir's sharing algorithm  $\mathcal{SH}$  picks  $a_1, \dots, a_t \xleftarrow{\$} F$  and defines the degree  $\leq t$  polynomial  $f: F \rightarrow F$  via

$$f(x) = s + \sum_{i=1}^t a_i x^i,$$

so that  $f(0) = s$ . The share of player  $i \in \{1, \dots, n\}$  is  $(i, f(e_i))$ .

This is a perfect  $(t, n)$ -secret sharing scheme.

The fact that a degree  $t$  polynomial is determined by any  $t + 1$  points on it allows recovery, which is done via the polynomial interpolation algorithm.

Privacy is because the values of  $f$  on any  $t$  distinct points are random and independent elements of  $F$ .

# Background on polynomials

For  $a_0, a_1, \dots, a_t \in F$ , define  $P_{a_0, a_1, \dots, a_t}: F \rightarrow F$  by

$$P_{a_0, a_1, \dots, a_t}(x) = \sum_{i=0}^t a_i x^i .$$

For  $R \subseteq \{1, \dots, n\}$  and  $i \in R$  define  $\delta_{R,i}: F \rightarrow F$  by

$$\delta_{R,i}(x) = \prod_{j \in R \setminus \{i\}} \frac{x - e_j}{e_i - e_j} .$$

Then for all  $\ell \in R$  we have

$$\delta_{R,i}(e_\ell) = \begin{cases} 1 & \text{if } \ell = i \\ 0 & \text{if } \ell \neq i . \end{cases}$$

Now for  $S = \{s_i : i \in R\} \subseteq F$  define  $Q_{R,S}: F \rightarrow F$  by

$$Q_{R,S}(x) = \sum_{i \in R} s_i \delta_{R,i}(x) .$$

Then  $Q_{R,S}(e_\ell) = s_\ell$  for all  $\ell \in R$ .

# Polynomial interpolation theorem

The polynomial interpolation theorem is the following.

Let  $F$  be a finite field of size at least  $n + 1$ . Fix some distinct points  $e_1, \dots, e_n \in F$ .

Let  $a_0, a_1, \dots, a_t \in F$  and let  $s_i \leftarrow P_{a_0, a_1, \dots, a_t}(e_i)$  for  $1 \leq i \leq n$ .

Let  $R \subseteq \{1, \dots, n\}$  have size  $|R| \geq t + 1$  and let  $S = \{s_i : i \in R\}$ .

Then  $Q_{R,S} = P_{a_0, a_1, \dots, a_t}$ .

In particular  $Q_{R,S}(0) = a_0$ .

# Shamir's $(t, n)$ perfect secret sharing scheme

Let  $F$  be a finite field of size at least  $n + 1$ . Fix some distinct points  $e_1, \dots, e_n \in F$ .

We define secret-sharing scheme  $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$  via

<b>Alg</b> $\mathcal{SH}(s)$ // $s \in F$	<b>Alg</b> $\mathcal{RE}(R, \{(i, s_i)\}_{i \in R})$
For $i = 1, \dots, t$ do $a_i \xleftarrow{\$} F$	$S \leftarrow \{s_i : i \in R\}$
For $i = 1, \dots, n$ do $s_i \leftarrow P_{s, a_1, \dots, a_t}(e_i)$	$s \leftarrow Q_{R, S}(0)$
return $\{(i, s_i)\}_{i=1}^n$	return $s$

This is a  $(t, n)$  perfect secret-sharing scheme.

Correctness, which assumes  $|R| \geq t + 1$ , follows from the polynomial interpolation theorem.

Perfect privacy is because any  $t$  of  $s_1, \dots, s_n$  are randomly and independently distributed over  $F$ .

## $n$ -party functionalities

A  $n$ -party functionality  $n\text{-}\mathcal{PF} = (\mathcal{I}, \mathcal{F})$  is a pair of algorithms that operate as follows:

- $(l_1, \dots, l_n) \xleftarrow{\$} \mathcal{I}$  — The initialization algorithm  $\mathcal{I}$  generates keys for parties  $1, \dots, n$  respectively.
- $(y_1, \dots, y_n) \leftarrow \mathcal{F}_{(l_1, \dots, l_n)}(x_1, \dots, x_n)$  — the functionality  $\mathcal{F}$  takes the inputs  $x_1, \dots, x_n$  of parties  $1, \dots, n$ , respectively, and (deterministically) returns outputs  $y_1, \dots, y_n$  for the parties, respectively.

We denote  $y_i$  by  $\mathcal{F}_{(l_1, \dots, l_n)}(x_1, \dots, x_n)[i]$ .

We say that  $n\text{-}\mathcal{PF}$  is keyless if  $l_1 = \dots = l_n = \varepsilon$ , in which case we may omit writing  $l_1, \dots, l_n$ .

# MPC protocols for $n$ -party functionalities

We assume each pair  $i, j$  of the  $n$  parties is connected by a secure (private and authenticated) channel  $i \rightarrow j$ . Sometimes a broadcast channel is also assumed.

Party  $P_i$  is initialized with  $I_i$  and has an input  $x_i$ . The parties exchange messages, in a sequence of rounds. At the end of the protocol,  $P_i$  has an output  $y_i$ .

The syntax of a protocol  $\Pi$  says how parties compute the messages they send each other, and how they determine their outputs.

Correctness asks that  $y_i = \mathcal{F}_{(I_1, \dots, I_n)}(x_1, \dots, x_n)[i]$  for  $1 \leq i \leq n$ .

A parameter  $t$  designates the maximum number of dishonest (also called bad or corrupted) parties that the protocol can tolerate.

Privacy, sometimes called  $t$ -privacy, then asks that any subset  $T \subseteq \{1, \dots, n\}$  of the parties of size  $|T| \leq t$  learns nothing about  $\{x_i : i \notin T\}$  other than what is revealed by  $\{\mathcal{F}_{(I_1, \dots, I_n)}(x_1, \dots, x_n)[i] : i \in T\}$ .

In the semi-honest (also called honest but curious) setting, the parties in  $T$  continue to follow the protocol with regard to what they compute and what messages they send.

In the malicious setting, the computation and messages sent by parties in  $T$  is controlled by the adversary.

We will only consider the semi-honest setting.

# The MPC universe

Defining privacy, and even correctness, is delicate, and the literature has many definitions. UC-security [Ca00] is a popular one.

There exist MPC protocols to securely compute any given  $n$ -party functionality, tolerating up to  $t < n/3$ , or even  $t < n/2$ , dishonest players.

These general protocols represent the functionality as a circuit and evaluate it gate by gate.

There is now a great deal of industry and research interest in MPC. There is a lot of work on efficient protocols, both general and for specific functionalities of interest.

Many software libraries. See [awesome mpc](#), [multiparty.org](#).

Many startups. Eg. [Unbound](#), [Baffle](#).

# Applications of MPC

Lindell divides applications into two classes.

The first class, *privacy*, is applications where the parties enter with their own private inputs.

In the second class, *security*, a single party has a value, like a secret cryptographic key, that they aim to secure by sharing it across  $n$  servers and then using MPC to compute the cryptographic function under the key.

In the following we will discuss one example in each category: Private Sum in the first and threshold RSA-FDH signatures in the second.

# Application stories for MPC

Some applications that one hears being discussed are the following.

Each party  $i \in \{1, \dots, n\}$  has a large dataset  $x_i$ .

They want to run a machine-learning (ML) algorithm on  $x_1, \dots, x_n$  to create a model that can later be queried.

But they don't want to share their dataset with other parties.

So an MPC protocol is run to create the model.

It could be healthcare or genomic data, and the parties could be hospitals or medical research organizations.

It could be data on user browsing or media-watching history, and the parties could be ...

# Private sum

Party  $i$  ( $1 \leq i \leq n$ ) has an integer  $x_i \in \mathbb{Z}_N$

The parties want to know the value of  $S = x_1 + \dots + x_n$

But party  $i$  does not wish to reveal  $x_i$  to the others.

Possible scenarios for this:

- $x_i \in \mathbb{Z}_{101}$  is the score of student  $i$  on quiz and  $S/n$  is the average
- $x_i \in \mathbb{Z}_{10^{15}}$  is the salary of employee  $i$  and  $S/n$  is the average
- $x_i \in \mathbb{Z}_2$  is the vote of voter  $i$  on a proposition and  $S$  is the tally.

We capture this by setting  $M = nN$  and asking for a  $(n-1)$ -private MPC protocol for the keyless  $n$ -party functionality  $n\text{-}\mathcal{PF} = (\mathcal{I}, \mathcal{F})$  in which

**Alg**  $\mathcal{F}(x_1, \dots, x_n)$  //  $x_1, \dots, x_n \in \mathbb{Z}_M$

$S \leftarrow (x_1 + \dots + x_n) \bmod M$  ; Return  $(S, \dots, S)$

When  $x_1, \dots, x_n \in \mathbb{Z}_N$  we have  $(x_1 + \dots + x_n) \bmod M = x_1 + \dots + x_n$  so  $\mathcal{F}$  returns the desired sum.

## How private can it get?

Suppose  $n = 4$ . At the end of the protocol, all parties learn  $S = (x_1 + x_2 + x_3 + x_4) \bmod M$ .

Party 1 knows  $x_1, S$  and thus can determine  $x_2 + x_3 + x_4 = (S - x_1) \bmod M$ .

If parties 1,2 collude they know  $x_1, x_2, S$  and can determine  $x_3 + x_4 = (S - x_1 - x_2) \bmod M$ .

If parties 1,2,3 collude they know  $x_1, x_2, x_3, S$  and can determine  $x_4 = (S - x_1 - x_2 - x_3) \bmod M$ .

Depending on context, the above losses of privacy may be significant. Agreeing to run the protocol means accepting and understanding this.

The privacy guarantee from the protocol is that no more than the above leaks, not that nothing leaks.

## Private sum protocol round 1: secret sharing

Let  $SS = (\mathcal{SH}, \mathcal{RE})$  be the above  $(n-1, n)$ -perfect secret sharing scheme with the group being  $Z_M$  under addition modulo  $M$ .

Each party  $i \in \{1, \dots, n\}$  does the following:

- generates shares of its input  $x_i \in Z_M$  via  $\{(j, x_{i,j})\}_{j=1}^n \stackrel{\$}{\leftarrow} \mathcal{SH}(x_i)$ .
- Sends  $x_{i,j}$  to party  $j$  ( $1 \leq j \leq n$ ) over the  $i \rightarrow j$  secure channel.

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

The modulo  $M$  sum of the elements of row  $i$  of the matrix is  $x_i$ .

Party  $j$  has the elements in column  $j$ .

The modulo  $M$  sum of all elements of the matrix is

$(x_1 + x_2 + x_3 + x_4) \bmod M$ .

## Private sum protocol rounds 2,3

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

Each party  $j \in \{1, \dots, n\}$  does the following:

- Receives  $x_{i,j}$  from party  $i$  in prior round.
- Lets  $c_j \leftarrow (x_{1,j} + x_{2,j} + \dots + x_{n,j}) \bmod M$
- Sends  $c_j$  to all parties  $i \in \{1, \dots, n\}$

Finally each party  $i \in \{1, \dots, n\}$  does the following:

- Receives  $c_j$  from party  $j$  in prior round.
- Lets  $S \leftarrow (c_1 + \dots + c_n) \bmod M$ .
- Outputs  $S$  and halts.

## Private sum protocol: correctness

$S$  is the modulo  $M$  sum of all entries in the matrix, which is the desired result.

In more detail, we know that

$$\text{— } x_i = \sum_{j=1}^n x_{i,j} \pmod{M} \text{ for } 1 \leq i \leq n.$$

$$\text{— } c_j = \sum_{i=1}^n x_{i,j} \pmod{M} \text{ for } 1 \leq j \leq n.$$

So

$$\begin{aligned} S &= \sum_{j=1}^n c_j \pmod{M} \\ &= \sum_{j=1}^n \sum_{i=1}^n x_{i,j} \pmod{M} \\ &= \sum_{i=1}^n \sum_{j=1}^n x_{i,j} \pmod{M} \\ &= \sum_{i=1}^n x_i \pmod{M} \\ &= \mathcal{F}(x_1, \dots, x_n). \end{aligned}$$

# Private sum protocol: Privacy

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

At the end of the protocol, party 1 knows the elements in the first row, the elements in the first column, and the column sums  $c_1, c_2, c_3, c_4$ .

What can it deduce about  $x_2, x_3, x_4$ ? We claim, nothing beyond what it knew a priori and  $x_2 + x_3 + x_4$ .

Similarly, we claim any set  $T$  of parties can determine nothing about  $\{x_i : i \notin T\}$  other than what they knew a priori and  $\sum_{i \notin T} x_i$ .

Making these claims precise needs careful formalization of privacy definitions, and then proofs, that we omit.

# Key Exposure and threshold cryptography

Cryptography (whether symmetric or asymmetric) relies on secret keys. The secret key  $sk$  needs to be stored on some system.

Systems are vulnerable to compromise via breaches, malware, insider attack, ...

This puts secret keys at risk of exposure.

An approach to mitigate this risk is to share the key  $sk$  across multiple servers via a  $(t, n)$ -secret sharing scheme  $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$ .

That is  $\{(i, sk_i)\} \stackrel{\$}{\leftarrow} \mathcal{SH}(sk)$  and server  $i$  holds  $sk_i$ .

Up to  $t$  servers can be compromised without exposing  $sk$ .

**Q:** But how do we perform the required cryptographic operations (signing, decryption, ...) when the key  $sk$  is shared across  $n$  servers?

**A:** Use a MPC protocol.

This is called threshold cryptography.

# Threshold signatures

The case where  $sk$  is the signing key for a signature scheme is of particular interest.

The signer may be a Certificate Authority (CA) that signs certificates under  $sk$ .

Exposure of  $sk$  would mean that certificates previously issued under it can no longer be trusted.

This could invalidate millions of certificates and significantly disrupt TLS-based Internet services.

Or the verification key  $vk$  corresponding to  $sk$  could be a Bitcoin address. Exposure of  $sk$  would mean that all the bitcoin money held under  $vk$  could be stolen.

Threshold signatures may be of interest for such reasons.

# Threshold RSA-FDH signatures setup

**Signer keys:** Signing key  $sk = (N, d)$  and verification key  $vk = (N, e)$ .

**Signatures:**  $S_{N,d}(M) = H(M)^d \bmod N$ .

We'll target threshold RSA-FDH for the case  $t = n - 1$ .

**Tool:** A  $(n - 1, n)$ -secret sharing scheme  $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$ , over the group  $Z_{\varphi(N)}$  with the group operation being addition modulo  $\varphi(N)$ .

**Sharing the key:** The signer, now party 1,

— Lets  $\{(i, d_i)\} \stackrel{\$}{\leftarrow} \mathcal{SH}(d)$ , and

— sends  $(N, d_i)$  to party  $i$  over the secure  $1 \rightarrow i$  channel ( $1 \leq i \leq n$ ).

The signer then deletes  $d$  from its system.

So we have:  $d = (d_1 + \dots + d_n) \bmod \varphi(N)$ .

Compromise of any  $n - 1$  parties does not allow adversary to obtain  $d$ .

# Threshold RSA-FDH signatures: naive protocol

Party  $i$  has  $(N, d_i)$ .

All parties have the message  $M$  to be signed.

Consider the following protocol:

- Party  $i$  sends  $d_i$  to party 1 over the  $i \rightarrow 1$  channel ( $1 \leq i \leq n$ ).
- Party 1 recovers  $d \leftarrow \mathcal{RE}(\{1, \dots, n\}, \{(i, d_i)\})$ .
- Party 1 computes signature  $x \leftarrow H(M)^d \bmod N$  and then deletes  $d$ .

**Bad:**  $d$  is present, even if ephemerally, on party 1's system, and can be exposed.

**Worse:** The recovery process  $d \leftarrow (d_1 + \dots + d_n) \bmod \varphi(N)$  requires that party 1 stores  $\varphi(N)$  permanently on its system. But exposure of  $\varphi(N)$  allows the adversary to factor  $N$  and obtain  $d$ .

# Threshold RSA-FDH signatures: the protocol

Party  $i$  has  $(N, d_i)$ .

All parties have the message  $M$  to be signed.

The protocol is:

- Party  $i$  lets  $x_i \leftarrow H(M)^{d_i} \bmod N$  ( $1 \leq i \leq n$ ).
- Party  $i$  sends  $x_i$  to party 1 over the  $i \rightarrow 1$  channel ( $1 \leq i \leq n$ ).
- Party 1 lets  $x \leftarrow \prod_{i=1}^n x_i \bmod N$ .
- Party 1 outputs  $x$  as the signature of  $M$ .

**Correctness:**

$$\begin{aligned}x &= \prod_{i=1}^n x_i \bmod N = \prod_{i=1}^n H(M)^{d_i} \bmod N \\ &= H(M)^{d_1 + \dots + d_n} \bmod N = H(M)^{(d_1 + \dots + d_n) \bmod \varphi(N)} \bmod N \\ &= H(M)^d \bmod N.\end{aligned}$$

# Threshold RSA-FDH: the functionality

Let  $\mathcal{K}_{\text{rsa}}$  be an RSA generator and define  $n$ -party functionality  $n\text{-}\mathcal{PF} = (\mathcal{I}, \mathcal{F})$  as follows:

Alg  $\mathcal{I}$

$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\text{rsa}}$   
 $\{(i, d_i)\}_{i=1}^n \xleftarrow{\$} \mathcal{SH}(d)$   
Return  $((N, d_1), \dots, (N, d_n))$

Alg  $\mathcal{F}_{((N, d_1), \dots, (N, d_n))}(M, \dots, M)$

$x \leftarrow H(M)^{d_1 + \dots + d_n} \bmod N$   
Return  $(x, \varepsilon, \dots, \varepsilon)$

Above  $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$  is the  $(n-1, n)$ -secret sharing scheme, over  $Z_{\varphi(N)}$  viewed as a group under addition modulo  $\varphi(N)$ .

The claim would be that the above protocol is a correct and  $t$ -private protocol for this  $n$ -party functionality.

# Threshold RSA-FDH: security

The adversary, given verification key  $(N, e)$ , can:

- Specify any set  $T \subseteq \{1, \dots, n\}$  of its choice with  $|T| \leq n - 1$ , and get  $d_i$  for all  $i \in T$ .
- Mount a chosen-message attack in which it requests signatures of messages  $M_1, \dots, M_q$  of its choice.

In response to signing request  $M_i$ , the protocol is executed on input  $M_i$ .

The adversary gets all random choices of parties in  $T$ , and communications to, or from, parties in  $T$ .

It also gets the final signature of  $M_i$ .

Security asks that after this the adversary is still unable to output a forgery, meaning a message  $M \notin \{M_1, \dots, M_q\}$  and a valid signature  $x$  of  $M$ .

Then the claim would be that the above protocol provides this security assuming one-wayness of the RSA generator  $\mathcal{K}_{\text{rsa}}$  underlying the scheme.

The following questions are rarely asked with regard to the technology that computer scientists develop, but should be asked more often:

**Who benefits?**

**What is the societal impact?**

We ask this in particular for MPC.

# Surveillance capitalism

Corporations collect data about us.

They run AI/ML algorithms to create models.

This is used to influence our behavior and choices.

It brings large profits for corporations.

This is called surveillance capitalism.

Surveillance capitalism has a negative impact on human and societal well-being and liberty.

# Surveillance capitalism

From [Wikipedia](#):

Surveillance capitalism is an economic system centered around the commodification of personal data with the core purpose of profit-making ... surveillance capitalism exploits and controls human nature ...

... collecting and processing data ... might present a danger to human liberty, autonomy, and well-being. Capitalism has become focused on expanding the proportion of social life that is open to data collection and data processing. This may come with significant implications for vulnerability and control of society as well as for privacy ...

See books, videos, movies; for example:

Shoshana Zuboff. [VPRO documenary, The Age of Surveillance Capitalism.](#)

Cathy O'Neil. [Weapons of Math Destruction.](#)

[The Social Dilemma, Coded Bias.](#)

# An MPC story

MPC is popularized via stories like the following.

Party  $i \in \{1, \dots, n\}$  has a data set  $x_i$ .

They want to compute a model  $f(x_1, \dots, x_n)$ , but party  $i$  does not want to reveal  $x_i$  to the other parties.

How can they do this? Via an MPC protocol.

In this view, MPC is a privacy providing enabler.

The story would be that the data is our (my, your) healthcare information.

We want this to be private, but we also want the cure for cancer, right?

In that view, MPC is a way to allow the parties to reach beneficial goals without compromising individual privacy.

# A different MPC story

Party  $i \in \{1, \dots, n\}$  has a data set  $x_i$ .

They want to compute a model  $f(x_1, \dots, x_n)$ , but party  $i$  does not want to reveal  $x_i$  to the other parties.

How can they do this? Via an MPC protocol.

I suggest an alternative viewpoint:

MPC (including FHE) is a way to **violate** privacy.

As such it can be used to support and extend surveillance capitalism.

These parties are not concerned with our privacy.

The reason they want to keep the data private is that the data has large financial value for them.

They want to realize this value and stay ahead of competitors.

MPC helps them do this.

## A different MPC story

As such, MPC can be used to support and extend surveillance capitalism.

Information sharing that may have otherwise been prevented by law, or the party's selfish interests, now becomes possible.

Models and the surveillance apparatus may potentially get more sophisticated and more damaging to individuals.

This is not to deny the existence of socially valuable uses of MPC or 2PC, such as Callisto, and PSM for Have I been Pwned.

But these are economically insignificant.

# The social context

Computer science education is almost exclusively technical.

What is taught, and learned, is how to make tools.

The dream of most students is to get jobs at the large corporations.

But these corporations practice surveillance capitalism.

Ask, what is the social impact of the technology we develop?

Ask, who does this help?

Does it help power, or does it help people?

Can we change course to help people instead of power?