Syllabus

Expected themes (subject to adaptation)

- Encryption and Zero-knowledge
- Verifiable Voting
- Anonymous Credentials
- Crypto Currencies
- Secure Two-Party Protocols
- Big Data and Statistical Privacy
- Mass Surveillance & Post-Snowden Cryptography

Supports

Available on the class website:

- Slides and/or notes
- Pointers to online references

General cryptography references:

- *Introduction to Modern Cryptography*
  by J. Katz and Y. Lindell – Chapman & Hall/CRC
- *A Graduate Course in Applied Cryptography*
  by D. Boneh and V. Shoup
  Draft available on `http://toc.cryptobook.us/`

Privacy Enhancing Technologies

O. Pereira, T. Peters and F.-X. Standaert

Zero-Knowledge Proofs

Zero-Knowledge Proofs – 1985

THE KNOWLEDGE COMPLEXITY OF INTERACTIVE PROOF SYSTEMS*

SHAFI GOLDWASSER†, SILVIO MICALI†, AND CHARLES RACKOFF‡

Abstract. Usually, a proof of a theorem contains more knowledge than the mere fact that the theorem is true. For instance, to prove that a graph is Hamiltonian it suffices to exhibit a Hamiltonian tour in it; however, this seems to contain more knowledge than the single bit Hamiltonian/non-Hamiltonian.

In this paper a computational complexity theory of the "knowledge" contained in a proof is developed. Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question. Examples of zero-knowledge proof systems are given for the languages of quadratic residuosity and quadratic nonresiduosity. These are the first examples of zero-knowledge proofs for languages not known to be efficiently recognizable.

Proofs

"Traditional" mathematical proofs:

"A list of reasons that shows a statement to be true"

- Non interactive
- No unique verifier in mind

Interactive proofs

Three ingredients:

1. A *prover* P, possibly unbounded in its power
2. A *verifier* V, must run efficiently (PPT)
3. A language $L \subset \{0,1\}^*$ defining a set of true statements

Motivations:

- Even if P is unbounded, he should not be able to prove wrong things
- V must be able to perform his task effciently
- L can be a lot of things:
  - set of integers that are the product of two primes
  - set of pairs of isomorphic graphs
  - set of true theorem statements

Interactive proofs

The pair (P, V) is an *interactive proof system* for L if it is:

1. **Complete**: If $\mathfrak{l} \in L$ then the probability that P does not convince V is negligible in $|\mathfrak{l}|$ When it's true, I can prove it
2. **Sound**: If $\mathfrak{l} \notin L$ then the probability that any P^* convinces V is negligible in $|\mathfrak{l}|$ When not true => P do whatever => V reject

Observations:

- Proofs are probabilistic
- V can be convinced even if P^* is unbounded
- Examples:
  - For the set of isomorphic graphs: send an isomorphism
  - For the bi-prime integers: send factors

Zero-knowledge proofs

Motivation:

- Protect the prover: the verifier should not learn anything but the fact that $\mathfrak{l} \in L$

On doit définir un transcript
simulator(V*,I) -> trans'
V* ne doit pas apprendre de nouvelle info par P. Sinon il triche et génère par lui même. V* serra convaincu par l'ordre du transcript (comme ASK and POST, TELL, SYN ...)
=> les INTERACTIONS sont importantes

Idea:

- Let *trans* be the transcript of the discussion between P and any PPT V^* on input $\mathfrak{l}$.
- It should be feasible to produce something indistinguishable from *trans* just from $\mathfrak{l}$

Observations:

- This "simulator" can build *trans* in any order!
  (So could the verifier if he tries to produce *trans*)
- No verifier can convince that a transcript is "real":
  He could have produced it himself!

Zero-knowledge proofs

(P, V) is a *perfect zero-knowledge* interactive proof system for L if $\forall$ PPT V^*, $\exists$ a PPT simulator $\mathcal{S}_{V^*}$ s.t. $\forall \mathcal{E}$: Environnement

$$Pr[\mathcal{E}(trans_{(P,V^*)}(\mathfrak{l})) = 1] = Pr[\mathcal{E}(trans_{\mathcal{S}_{V^*}}(\mathfrak{l})) = 1]$$

Env reçois un real transcrit

Env reçois un transcrit produit par le simulateur sur le même input

where:

- $trans_{(P,V^*)}(\mathfrak{l})$ is the transcript of the interaction of P and V^* on input $\mathfrak{l}$
- $trans_{\mathcal{S}_{V^*}}(\mathfrak{l})$ is the output of $\mathcal{S}_{V^*}$ on input $\mathfrak{l}$
- $\mathcal{E}$ is anyone who tries to distinguish the two transcripts

Remark:

- One could define *computational zero-knowledge*:
  - $\mathcal{E}$ must be PPT
  - the probabilities can have a negligible difference

Discrete Log Statements

g = generator/primitive generator. Il est equally distributed (le tour de l'horloge)

Order Zq = |Zq| = |{0;q-1}|

Our focus in this class:

ZK proofs for DL statements

The Discrete Logarithm (DL) problem:

1. Take a finite group $\mathbb{G}$ such that $|\mathbb{G}| = q$ and $g \in \mathbb{G}$
2. Pick $x \leftarrow \mathbb{Z}_q$, compute $h = g^x$
3. Give $\mathcal{A}$: $(\mathbb{G}, g, h)$
4. $\mathcal{A}$ wins if he finds $x = \log_g(h)$

This problem is believed to be hard in well chosen large groups

The group $\mathbb{Z}_p^*$

Let p be a large prime integer
We define:

- $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$, equipped with
- multiplication $\bmod p$ as operation

Example:

- $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$
- $3 \times 4 = 5$ in $\mathbb{Z}_7^*$

This is a group when p is prime:

- The product of any two elements of $\mathbb{Z}_p^*$ is in $\mathbb{Z}_p^*$
- 1 is the neutral element $(1 \times a = a \times 1 = a)$
- it is associative $(a \times (b \times c) = (a \times b) \times c)$
- every element has a multiplicative inverse
  $(2 = 4^{-1}, 3 = 5^{-1}, 6 = 6^{-1}$ in $\mathbb{Z}_7^*)$
  N.B.: when p is not prime, some elements in $\{1, 2, \ldots, p-1\}$ do not have a multiplicative inverse

The group $\mathbb{Z}_p^*$

Let $p = 47$ and $g = 17$
The powers of 17 are:
$[17, 7, 25, 2, 34, 14, 3, 4, 21, 28, 6, 8, 42, 9, 12, 16, 37, 18, 24, 32, 27, 36, 1]$

attention au cycle sinon je problème devient faisable

A DL problem:

- When given $\mathbb{Z}_{47}^*$, $g = 17$ and $h = 16$,
- Can $\mathcal{A}$ find $x : 16 = 17^x$?

For the DL to be hard, we need the sequence of powers of g to be long before cycling

$\mathbb{Z}_p^*$ is a cyclic group when p is prime [Gauss, 1801]:

- When p is prime, there are g values that *generate* the full $\mathbb{Z}_p^*$:

$$\langle g \rangle = \{g, g^2, \ldots, g^{p-1} = 1\} = \mathbb{Z}_p^*$$

- We call such g values *generators* of $\mathbb{Z}_p^*$

The group $\mathbb{Z}_p^*$

RFC 3526 MODP Diffie-Hellman groups for IKE May 2003

4. 3072-bit MODP Group

This group is assigned id 15.

STANDART PRIME ==> This prime is: 2^3072 - 2^3008 - 1 + 2^64 * { [2^2942 pi] + 1690314 }

Its hexadecimal value is:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AAAC42D AD33170D 04507A33 A85521AB DF1CBA64
ECFB8504 58DBEF0A 8AEA7157 5D060C7D B3970F85 A6E1E4C7
ABF5AE8C DB0933D7 1E8C94E0 4A25619D CEE3D226 1AD2EE6B
F12FFA06 D98A0864 D8760273 3EC86A64 521F2B18 177B200C
BBE11757 7A615D6C 770988C0 BAD946E2 08E24FA0 74E5AB31
43DB5BFC E0FD108E 4B82D120 A93AD2CA FFFFFFFF FFFFFFFF
```

The generator is: 2.

Proof for Discrete Logarithm

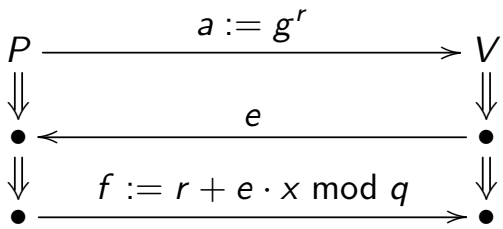
Solving the identification based on the DL problem:

- P is identified by (g, h) and knows $x : h = g^x$
- How can P demonstrate knowledge of x in ZK?

Schnorr's protocol [1988]

Let $\mathbb{G}$ be a group of prime order q and $g \in \mathbb{G}$

P prend un r, envoie a
V défi P avec e
P prouve en calculant f, sans révéler x
V vérifie (que P connait X)

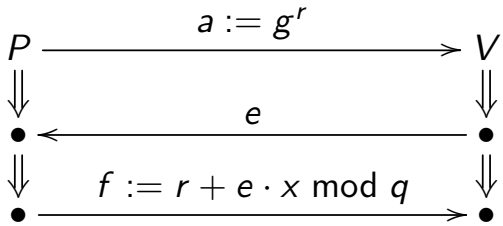


secret: x
G: groupe de premier ordre q et générateur g
clé publique : h = g^x mod q
r: nb random de Zq
e: défi (de Zq, pour Soundness)
f = r +e.x mod q

P proves knowledge of x to V who has $h := g^x$

1. P chooses $r \leftarrow \mathbb{Z}_q$ and commits through $a := g^r$
2. V challenges with a random $e \leftarrow \mathbb{Z}_{2^n}$ séquence de N bits :=> 2^N possibilités (pour Soundness)
3. P responds with $f := r + e \cdot x \bmod q$
4. V accepts if $g^f = a \cdot h^e = g^r \cdot (g^x)^e$

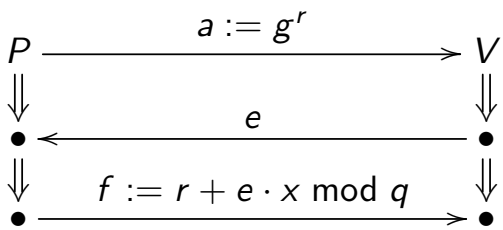
Schnorr's protocol



Completeness: obvious
Soundness:

- In order to reply with non-negligible probability, P must be able to respond to more than 2 challenges, say e and e'
- Suppose that P can produce transcripts (a, e, f) and (a, e', f') s.t. $g^f = a \cdot h^e$ and $g^{f'} = a \cdot h^{e'}$
- Then $g^f/(g^x)^e = g^{f'}/(g^x)^{e'}$ and $x = \frac{f-f'}{e-e'}$

Schnorr's protocol



avec un grand e on ne peut plus retry si facilement (simulateur non efficient)

Honest verifier zero-knowledge:
that is, ZK only when V follows the protocol
The simulator can:

- Choose e, f at random and compute $a := g^f/(g^x)^e$

These a, e, f follow the distribution of a real transcript:

- e is random and independent of the rest, as expected
- a and f are in one-to-one correspondence

This simulator does not work if, say, V sets e as the first n bits of a

- nobody knows if this protocol is "full" ZK

Σ-protocols

Π is a Σ-protocol for a relation R if:

- It is a 3-move protocol with completeness, made of a

a=> *commitment*, followed by a random *challenge*, and ending with a *response*

Soundness =>

- For any pair (a, e, f) and (a, e', f') of accepting conversations on input $\mathfrak{l}$ where $e \neq e'$, one can efficiently compute $\mathfrak{w} : (\mathfrak{l}, \mathfrak{w}) \in R$
- There is an efficient simulator that, on input $\mathfrak{l}$, e, produces (a, f) such that (a, e, f) is distributed as in a normal proof.

Not just proof that $\mathfrak{l} \in L = \{\mathfrak{l} : \exists \mathfrak{w} \text{ s.t. } (\mathfrak{l}, \mathfrak{w}) \in R\}$, but
proof of knowledge of a witness $\mathfrak{w} : (\mathfrak{l}, \mathfrak{w}) \in R$.

Privacy Enhancing Technologies

O. Pereira, T. Peters and F.-X. Standaert

Voting

Voc:
Tally = décompte (des votes)
Talliers = les gens qui font le décompte
Booth = isoloir

Application of ZK

Verifiable Elections – 1981

Technical Note
Programming Techniques and Data Structures
R. Rivest Editor

Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms

David L. Chaum
University of California, Berkeley

The technique can also be used to form rosters of untraceable digital pseudonyms from selected applications. Applicants retain the exclusive ability to form digital signatures corresponding to their pseudonyms. Elections in which any interested party can verify that the ballots have been properly counted are possible if anonymously mailed ballots are signed with pseudonyms from a roster of registered voters.

Verifiable Elections – 1987

Verifiable Secret-Ballot Elections

Josh Daniel Cohen Benaloh

YALEU/DCS/TR-561
September 1987

This thesis describes a practical scheme for conducting secret-ballot elections in which the outcome of an election is verifiable by all participants and even by non-participating observers. All communications are public, yet under a suitable number-theoretic assumption, the privacy of votes remains intact.

Elections

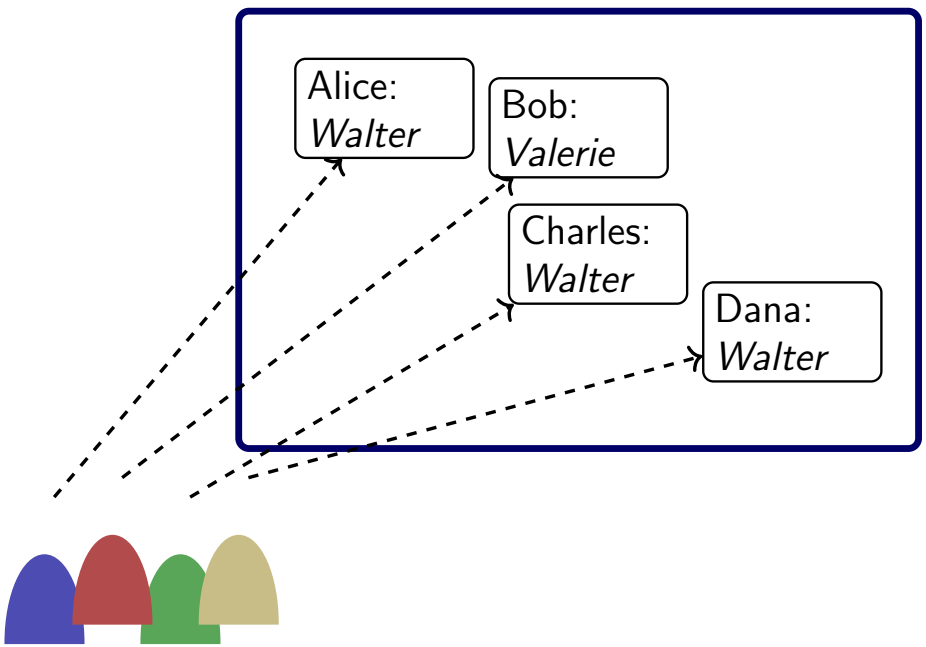
A simple protocol problem:

- compute the sum of n secret inputs

What makes it hard?

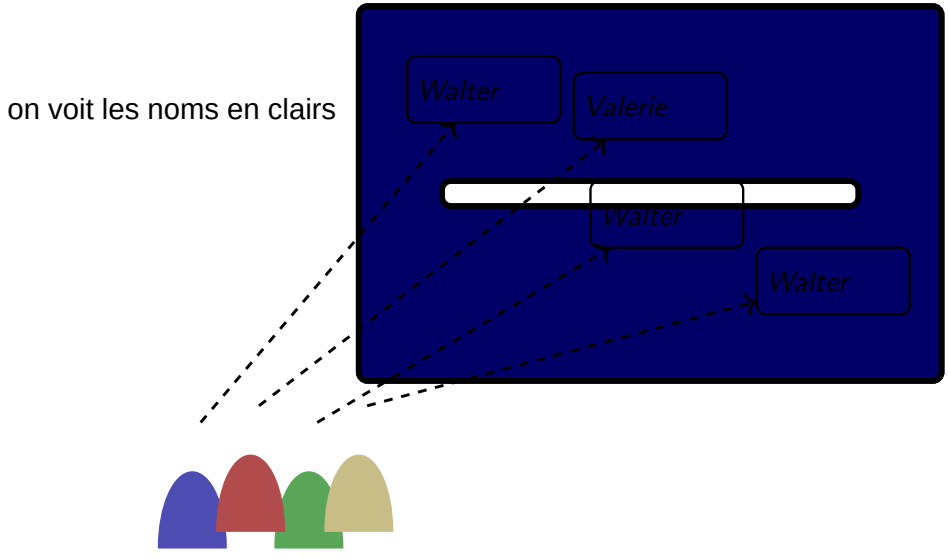
1. Election organizers have their own preferences regarding the results
2. Election organizers' computers may have been hacked
3. For Internet voting: $n \geq 1000000$ voters cannot be assumed to have honest devices running correct software securely

Open Elections



- Every voter can verify that nobody tampered with her/his vote
- Every voter can compute the tally
- No privacy, no coercion-resistance, no fairness, ...

A traditional paper approach



- With voting booth: privacy, coercion-resistance, fairness, ...
- **If** a voter keeps an eye on the ballot box and tally operations all day long,
  he can be convinced that:
  - his vote is untampered
  - the tally is based on valid votes and correct
- A minute of inattention is enough to break this

Verifiable Elections

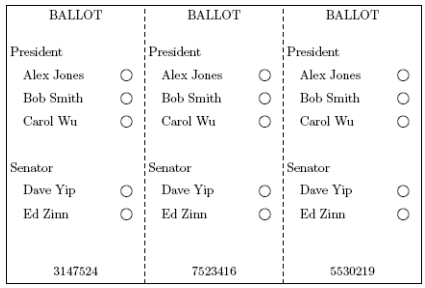
What do we want?

1. Correct results – primary goal of an election
2. Evidence of correct results
   Result can be verified even if:
   - All devices used in the election are malicious
     Software independence
   - All election officials are malicious
     Evidences should not (only) rely on a chain of custody

Example: ThreeBallot

BALLOT	BALLOT	BALLOT
President	President	President
Alex Jones ○	Alex Jones ○	Alex Jones ○
Bob Smith ○	Bob Smith ○	Bob Smith ○
Carol Wu ○	Carol Wu ○	Carol Wu ○
Senator	Senator	Senator
Dave Yip ○	Dave Yip ○	Dave Yip ○
Ed Zinn ○	Ed Zinn ○	Ed Zinn ○
3147524	7523416	5530219

Example: ThreeBallot



Election:

1. Print ballots:
   - 3 copies of candidate list
   - 3 random numbers at bottom
     secret (scratch-off layer, . . .)
2. Voter:
   - picks a paper ballot, goes to voting booth
   - picks candidates: 1 bullet = no, 2 bullets = yes
   - split ballot in 3 pieces, keep copy of one of them
   - puts all 3 pieces in the ballot box

on imprime une NOUVELLE copie

Example: ThreeBallot

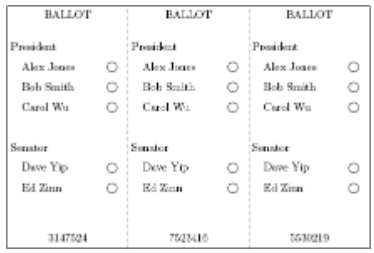
Tally:

1. Talliers:
   - Ballots are shuffled and scanned
   - Ballots are posted on a web bulletin board
   - For $3n$ ballot pieces, candidate gets $\sum$ of her bullets $-n$
2. Audit:
   - Anyone can recompute tally from bulletin board
   - Any voter can check that her/his ballot part copy is there

Does it work?

- Cheater does not know which ballot components are receipt
  Pr[modifying i ballots without detection] $= (2/3)^i$
  (Decrease remains exponential if x% of the voters verify)
- Receipt does not help a voter to prove how he voted
  (*Receipt freeness*)

Example: ThreeBallot

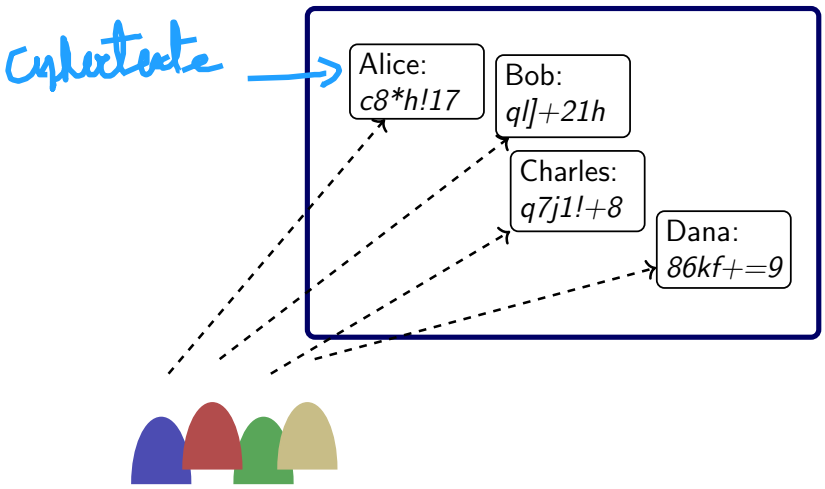


Limitations:

- Voters could fill all 3 bullets for a candidate
  Voters could fill no bullet for a candidate
- When many candidates, ballot reconstruction may be possible
- Receipt leaks statistical information about vote
  Good estimate of election result can be obtained just from receipts
- Difficulty to keep the random printed numbers secret

eg pour vote recteur

Verifiability from Cryptography



Keep public bulletin board but encrypt votes

- How to guarantee confidentiality towards talliers?
- How to compute the election result?
- How to be sure that ballots are valid and results correct?

How to encrypt votes?

Distributed homomorphic public key encryption

- Public key encryption
  - Votes are encrypted with a *public* encryption key *pk*
  - Tally is decrypted with a *secret* decryption key
- Homomorphism avoids decrypting individual votes:

$$\mathrm{Enc}_{pk}(v_1) \times \mathrm{Enc}_{pk}(v_2) = \mathrm{Enc}(v_1 + v_2)$$

Encrypt "yes" as $\mathrm{Enc}_{pk}(1)$ and "no" as $\mathrm{Enc}_{pk}(0)$
Then decrypt the product of all encrypted votes

- Distributed encryption
  - No single party ever holds the secret decryption key
  - Decryption is a distributed operation by a set of trustees

le produit de 2 enc == l'enc de leurs sommes

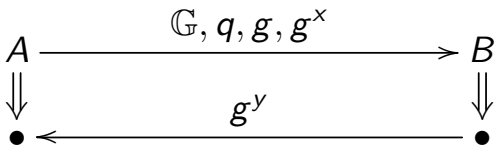
(pour compute le tally)

Towards public key encryption in DL groups

Générer une clé partagée (example avec les couleurs qui se mélangent ou les petits robots)

The Diffie-Hellman key exchange:

- Use cyclic group $\mathbb{G}$ of order q generated by g
- A and B select x and y uniformly at random in $\mathbb{Z}_q$

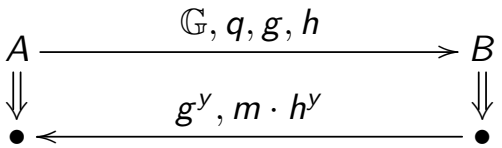


- Alice and Bob can compute $g^{xy} = (g^x)^y = (g^y)^x$
- A 3rd party able to extract a DL can extract x from (g, g^x) and derive $g^{xy} = (g^y)^x$
- We do not know any other way to compute g^{xy} in most groups

ElGamal Encryption

Ideas:

- Use $(\mathbb{G}, q, g, h = g^x)$ as public key and x as secret key
- Use $h^y = g^{xy}$ to hide $m \in \mathbb{G}$



- Decryption: $m = (m \cdot h^y)/(g^y)^x$

1) Key generation
q=large prime | g = generator | x = private key
h = g^x mod q = Enc(x)
2)Encryption
m = plainetext message
y = private key
c1 = g^y mod q
c2 = m.h^y
3)Decryption
m = c2/(g^y)^x = g^m = log g (g^m) = m

This is secure under the Decisional Diffie-Hellman (DDH) assumption

- DDH: It is hard to distinguish (g, h, g^y, h^y) from (g, h, g^y, g^z) with a random $z \in \mathbb{Z}_q$ (and $h = g^x$ for a random $x \in \mathbb{Z}_q$)
- For an ElGamal observer:

$$(g, h, g^y, m \cdot h^y) \approx (g, h, g^y, m \cdot g^z) \equiv (g, h, g^y, g^z)$$

So, given the public key, a ciphertext looks like a pair of random group elements, carrying no information about m

Exponential ElGamal

Exponential ElGamal: represent $m \in \{0,1\}$ as $g^m \in \mathbb{G}$

- Generate keys (pk, sk) as:
  - $sk = x$ with $x \leftarrow \mathbb{Z}_q$ and $|q| = n$
  - $pk = (\mathbb{G}, g, q, h = g^x)$
- $\mathrm{Enc}_{pk}(m) = (g^r, g^m h^r)$ with $r \leftarrow \mathbb{Z}_q$
- $\mathrm{Dec}_{sk}(c_1, c_2) = \log_g(c_2/c_1^x)$ – works if m is small enough

These ciphertexts are (additively) homomorphic!

- Encrypt 0/1 "counters" for each candidate
- Product of encrypted counters of each candidate is an encryption of number of preferences received

Distributed key generation for ElGamal

Run key generation among trustees $T_1, T_2, \ldots, T_t$:

1. Each T_i picks an ElGamal secret key x_i and publishes $h_i = g^{x_i}$
2. Everyone can compute $h = \prod h_i = g^{\sum x_i}$
   Encryption works as usual from $pk = (\mathbb{G}, q, g, h)$

Decryption of ciphertext $\mathrm{Enc}_{pk}(m) = (c_1, c_2)$:

1. Each T_i publishes $d_i = c_1^{x_i}$
2. Everyone can compute $g^m = c_2 / \prod d_i = c_2 / c_1^{\sum x_i}$

Distributed key generation for ElGamal

Is that good enough?

Getting $t-1$ decryption factors does not help:

still has hard to decrypt; you need ALL trustuees

- Suppose $x_1, \ldots, x_{t-1}$ are compromised
  We can remove them from ciphertexts
- Remaining ciphertext is $(g^r, g^m g^{x_t r})$
  But this is an encryption with public key g^{x_t}

Is key generation always ok?

- Suppose T_t picks x and computes his public key as $g^{x_t} = g^x / \prod_{i=1}^{t-1} g^{x_i} \Rightarrow T_t$ ignores x_t
- Now, public key is computed as $\prod_{i=1}^{t} g^{x_i} = g^x$

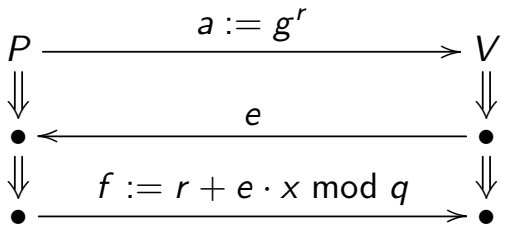
avec le dernier T, he can decrypt everything

Solution:

- Require each T_i to offer a Schnorr ZK proof that they know x_i

Schnorr's protocol

Talliers must prove that they know the secret key corresponding to their public key



P proves knowledge of x to V who has $(g, h := g^x)$

1. P chooses $r \leftarrow \mathbb{Z}_q$ and commits through g^r
2. V challenges with a random $e \leftarrow \mathbb{Z}_{2^n}$
3. P responds with $f := r + e \cdot x \bmod q$
4. V accepts if $g^f = a \cdot h^e \quad (= g^r \cdot (g^x)^e)$

Non-interactive (NI) ZK

Σ-protocols, including Schnorr, are interactive

- Each T_i would need to run Schnorr with every other T_j :(

Other option: make Σ protocols **non-interactive**

- Suppose we have a "random" function $\mathcal{H}$
  - $\mathcal{H}(a)$ looks random in $\mathbb{Z}_{2^n}$ for every new a
- Then P can compute $e := \mathcal{H}((g,h),a)$ and send *non-interactive* proof (a,e,f)
- $\mathcal{H}$ "emulates" the interaction:
  - P needs (g,h) and a before computing e
  - V can verify the value of e once it gets (g,h) and a
- Now, each T_i can compute and send *one* NI Schnorr proof

Such a random behaving function $\mathcal{H}$ is modeled as a *random oracle*
$\mathcal{H}$ is often realized with the SHA256 hash function or HMAC-SHA256

Distributed key generation for ElGamal

Run key generation among trustees $T_1, T_2, \ldots, T_t$:

1. Each T_i picks an ElGamal secret key x_i and publishes $h_i = g^{x_i}$
2. Each T_i selects $r_i \leftarrow \mathbb{Z}_q$, computes

$$a_i = g^{r_i},\ e_i = \mathcal{H}(g, h_i, a_i),\ f_i = r_i + e_i x_i$$

   and publishes a_i, e_i, f_i
3. Everyone can verify that $e_i = \mathcal{H}(g, h_i, a_i)$ and $g^{f_i} = a_i h_i^{e_i}$ and compute $h = \prod h_i = g^{\sum x_i}$
   Encryption works as usual from $pk = (\mathbb{G}, q, g, h)$

Decryption of ciphertext $\mathrm{Enc}_{pk}(m) = (c_1, c_2)$:

1. Each T_i publishes $d_i = c_1^{x_i}$
2. Everyone can compute $g^m = c_2 / \prod d_i = c_2 / c_1^{\sum x_i}$

How to be sure that the tally is correct?

Let (c_1, c_2) be the product of all counter ciphertext for a candidate

- Suppose T_1 gives decryption factor $c_1^{x_1 - 1}$
- Product of decryption factors is $g^{r(\sum x_i - 1)}$
- Decryption gives $g^m \cdot g^r$ instead of g^m

Diagnostic:

- Decryption will not succeed: DL of g^{m+r} probably cannot be extracted. But who is guilty?
- What if last decrypting trustee cheats when he dislikes the result?

Let us require trustees to *prove* correct decryption!

The Chaum-Pedersen Protocol

"2 Shnoor en même temps

Statement:

- d is a decryption factor for (c_1, c_2) w.r.t. public key h

Witness:

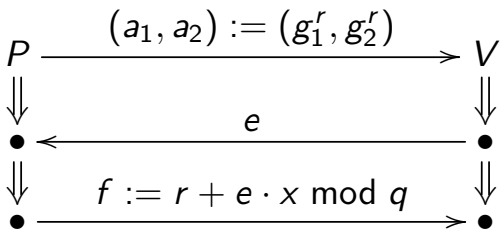
- witness: private key x (or x_i for distributed ElGamal)

Reformulation:
Prove that there is x such that:

- $h = g^x$ and $d = c_1^x$

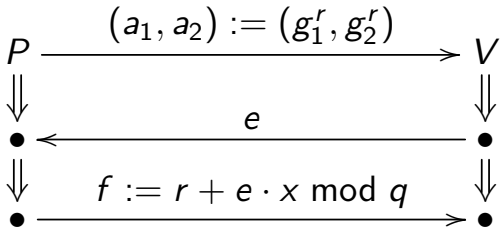
The Chaum-Pedersen protocol

P proves that he knows x such that $h_1 = g_1^x$ and $h_2 = g_2^x$



1. P chooses $r \leftarrow \mathbb{Z}_q$ and commits through g_1^r, g_2^r
2. V challenges with a random $e \leftarrow \mathbb{Z}_{2^n}$
3. P responds with $f := r + e \cdot x \bmod q$
4. V accepts if $g_1^f = a_1 \cdot (h_1)^e$ and $g_2^f = a_2 \cdot (h_2)^e$

The Chaum-Pedersen protocol



Completeness: Again, if P knows x and runs the protocol, V accepts

Soundness:

- If P can prove with $((a_1, a_2), e, f)$ and $((a_1, a_2), e', f')$ then $h_1 = g_1^x$ and $h_2 = g_2^x$ for $x = \frac{f-f'}{e-e'}$

Honest verifier zero-knowledge:

- Choose e, f at random and compute $a_1 := g_1^f/(h_1)^e$ and $a_2 := g_2^f/(h_2)^e$

NI zero-knowledge:

- Compute $e = \mathcal{H}(g_1, h_1, g_2, h_2, a_1, a_2)$

How to be sure that the ballots are valid?

A malicious voter could encrypt 100 instead of 0 or 1 counter

- Adds 100 votes for one candidate
- And he could encrypt -99 for another candidate

Diagnostic:

- Since we never decrypt individual votes, we need to have another way of knowing their validity

Let us require voters to add a proof that they encrypt 0 or 1!

Proving OR statements

Suppose we have:

- a Σ-protocol Π_0 for proving knowledge of w_0 s.t. $(x_0, w_0) \in R_0$
- a Σ-protocol Π_1 for proving knowledge of w_1 s.t. $(x_1, w_1) \in R_1$

Combining proofs:

- Proving knowledge of both w_0 and w_1 is easy: make 2 proofs
- Can we prove knowledge of either w_0 or w_1?

Applications:

- Prove that a ciphertext $c = x_0 = x_1$ encrypts 0 or 1 with randomness $w_0 = w_1$
  - Π_0 will be a proof that c encrypts 0
  - Π_1 will be a proof that c encrypts 1

  Of course, at most one of the two statements can be true
- Prove that I know one of the DL of $(h_1, \ldots, h_n)$ in base g (anonymous authentication)

Disjunctive proofs [CDS94]

Strategy:

1. P starts Π_0 and Π_1 in parallel
2. V gives a single challenge e
3. P provides two sub-challenges e_0, e_1 s.t. $e_0 + e_1 = e$ and two responses f_0 and f_1 consistent with e_0 and e_1
4. P can then choose one of e_0 or e_1 in advance, simulate the corresponding proof, and run the other proof honestly

Suppose prover has $w_i : (x_i, w_i) \in R_i$ (but not w_{1-i})

1. P selects random e_{1-i} and runs the simulator $\mathcal{S}_{1-i}$ to get a proof $(a_{1-i}, e_{1-i}, f_{1-i})$
2. P selects a_i as from Π_i's definition
3. P commits on (a_0, a_1) to V
4. V challenges with e
5. P computes $e_i = e - e_{1-i} \bmod 2^n$ and f_i from (w_i, a_i, e_i)
6. V accepts if (a_0, e_0, f_0) and (a_1, e_1, f_1) check for Π_0 and Π_1 and $e_0 + e_1 = e \bmod 2^n$

Proof that a ciphertext encrypts 0 or 1

How to prove that $c = (c_0, c_1) = (g^r, g^m h^r)$ encrypts m?

- Prove that $(c_0, c_1/g^m)$ encrypts 0
- Prove that I know r such that $c_0 = g^r$ and $c_1/g^m = h^r$ using the Chaum-Pedersen protocol

How to prove that $c = (c_0, c_1) = (g^r, g^m h^r)$ encrypts 0 or 1?
Suppose that c is an encryption of 0:

1. P runs simulator that c encrypts 1 gives (a_1, e_1, f_1)
2. P pick $r_0 \in \mathbb{Z}_q$ and compute $a_0 = (g^s, h^s)$
3. P sends (a_0, a_1) to V and obtains e
4. P computes $e_0 = e - e_1 \bmod 2^n$ and $f_0 = s + er$
5. P sends (e_0, e_1) and (f_0, f_1) to V
6. V checks that:
   - $e = e_0 + e_1$,
   - (a_0, e_0, f_0) is a proof that c encrypts 0, and
   - (a_1, e_1, f_1) is a proof that c encrypts 1

Putting it all together

Voting operations:

1. Trustees generate an ElGamal key pair in a distributed way
   Trustees publish a Schnorr proof that they know their secret key
2. Voters encrypt their choices as 0 or 1 for each candidate
   Voters prove the validity of the ciphertexts using 0-1 proofs
   Voters publish these ciphertexts and proofs
3. Anyone can multiply all the ciphertexts, candidate per candidate
   Homomorphic property means we have encryptions of tallies per candidate
4. Trustees publish decryption factors for each encrypted tally
   Trustees publish a Chaum-Pedersen proof that these factors are correct
5. Voters check that their encrypted vote is included in the tally
   Everyone can check that the encrypted votes are valid and that the announced tally matches the encrypted votes

Putting it all together: Helios ≥ 2.0

Helios ≥ 2.0 `https://vote.heliosvoting.org/`:

- Internet voting
- End-to-end verifiable: election results can be verified without requiring trust in any individual/software/hardware
- Designed for 2009 UCLouvain rector election
  First real-world large scale use of E2E verifiable system
- Used by ACM, IACR, bitcoin foundation, various universities, associations, boards, private companies, ...

Security model:

- Internet voting: coercion is out-of-scope
  (But we authorize re-voting)
- Voters are practically assumed to have trustworthy computers
  (Can detect corrupted devices, but cumbersome w.r.t. usability)

Putting it all together: Helios ≥ 2.0

Helios server:

- hosts election description, public keys, encrypted ballots, election results
- serves a voting booth as a web application
- serves public election bulletin board and audit tools
- serves admin tools supporting election creation, key generation, decryption, ...

Helios voting booth:

- Single page web application
- ElGamal Encryption and all the ZK proofs from the previous slides implemented in JavaScript
- Gives the voter a hash of their encrypted vote for tracking
- Not mandatory use: anyone can post a ballot using the server API

Putting it all together: Helios ≥ 2.0

Correctness guarantees:

- Bulletin board shows voters, encrypted votes, results and proofs
  - List of voters is public
  - Vote validity guaranteed by disjunctive proofs
  - Valid decryption of tally guaranteed by Chaum-Pedersen proofs
- Benaloh challenge can detect corrupted voting devices:
  1. Voting client commits on an encrypted vote
  2. Voter chooses to cast or audit

  If audit is chosen, voting device must release ciphertext randomness, and voter can verify on another device (and revote)

Privacy guarantees:

- All trustees must collaborate for any decryption operation
  ⇒ As long as one is honest, only election result will be decrypted

Putting it all together: Helios ≥ 2.0

Does it work?

- Millions of ballots tallied
- Voters like checking their vote on the bulletin board
- Working with encrypted data makes it easy and safe to monitor elections closely
  Easy detection and recovery from email management errors, vulnerability in authentication system, voter complaints, ...

Limitations?

- People just trust the server: few independent ballot preparation systems, bulletin boards, audit tools, ... are used
- Hardly deals with corrupted voting client threat

What about government elections?

What does it change?
State-sponsored attacks are in-scope

- DDoS on voting servers
  Costly infrastructure, used < 1 day/year
- Targeted malwares are realistic
  - Academic demo: pdf of candidate program triggering an Adobe Reader exploit, used to corrupt browser, which fakes voting website display
  - Real-world demo: 0-day on TLS used to transmit fake voting booth [Internet Election in NSW, Australia, 2015]
  - ...

What about government elections?

ElectionGuard `https://www.electionguard.vote/`

- Open-source SDK offering the protocols we discussed and more
- Motivation: be able to offer evidence that election outcome is correct
- Has been integrated in paper based voting systems
  1. Voting machines prints a human-readable ballot and ballot tracker and keeps all the ciphertexts and proofs in memory for verifiable tally,
  2. Hand-marked paper ballots are scanned, interpreted and encrypted on-the-fly for verifiable tally
- Deployments in the US and elsewhere: 2022 US mid-terms, municipal elections, some Internet-based elections, ...

Privacy Enhancing Technologies

O. Pereira, T. Peters and F.-X. Standaert

Anonymous Credential

signature, un user demande une signature à un prover, qui permet à user de montrer qu'il "connait" x (clé privée du prover) sans en fait l'avoir, grace à la signature qu'il a

Anonymous Credential – 1985

SECURITY WITHOUT IDENTIFICATION: TRANSACTION SYSTEMS TO MAKE BIG BROTHER OBSOLETE

Communications of the ACM *October 1985 Volume 28 Number 10*

The large-scale automated transaction systems of the near future can be designed to protect the privacy and maintain the security of both individuals and organizations.

"Big Brother"?

1st edition cover – 1949!

Security Without Identification

What could it mean?

> "*an individual uses a different account number or 'digital pseudonym' with each organization*"
>
> "*individuals keep secret keys from organizations and organizations devise other secret keys that are kept from individuals*"
>
> – David Chaum

Security Without Identification

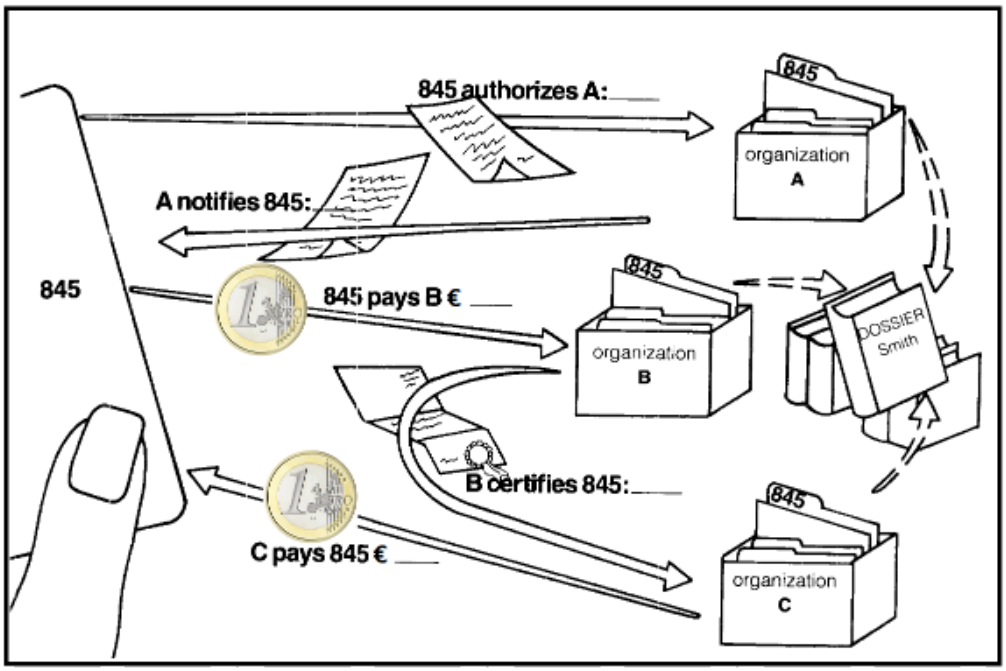
> "*an individual uses a different account number or 'digital pseudonym' with each organization*"

Should you be explicitly identified in all your (personal) electronic exchanges?

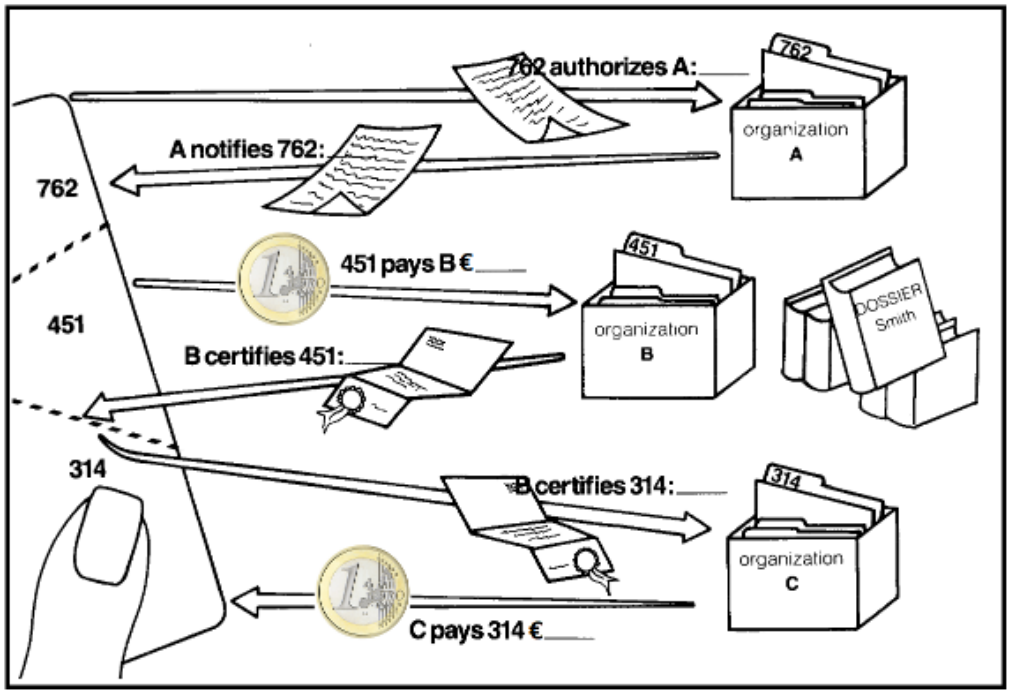
> "*individuals keep secret keys from organizations and organizations devise other secret keys that are kept from individuals*"

Cryptography can prevent both users and organizations from abusing each other

Protecting Users' Privacy?



Unlinkable Transactions



Anonymity as Unlinkability

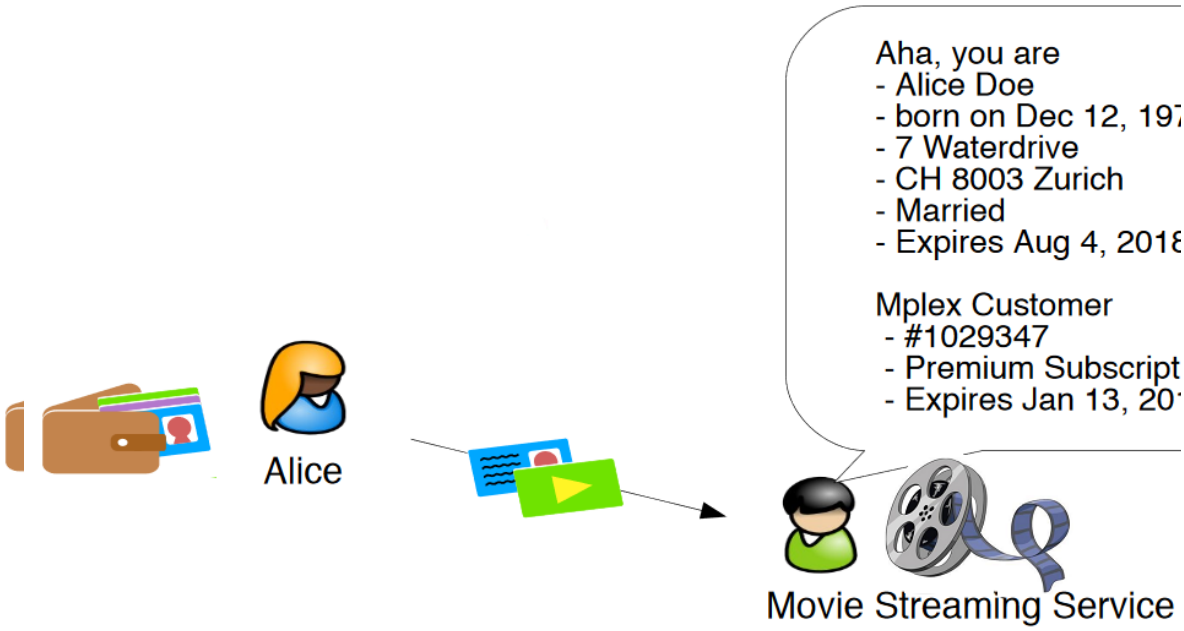
How to define this security notion?

- What if 451 is the only pseudonym getting certification from **B**?
- If **C** and **B** talk to each other they will learn $451 \sim 314$
- Real issue? Many many transactions in practice

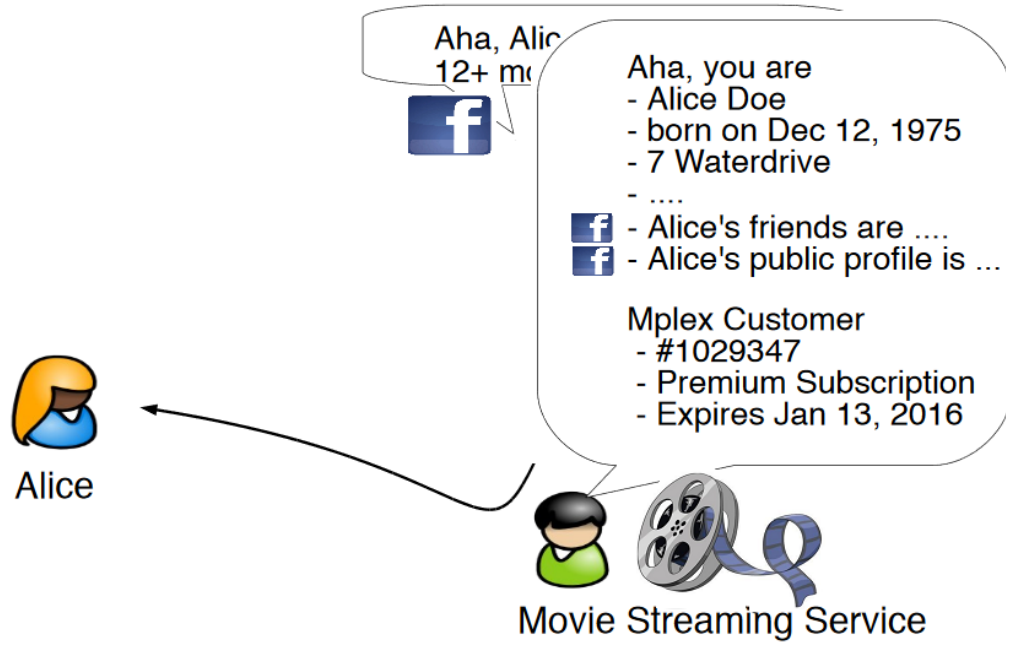
Extracting links should be hard: unfeasible to guess b

- assuming users U_0 and U_1 got certifications from **B**...
- assuming $U_{b \leftarrow \{0,1\}}$ made a transaction with **C**...
- ...even if the *adversary* corrupted **B** and **C**
- ...even if the *adversary* chose users U_0 and U_1

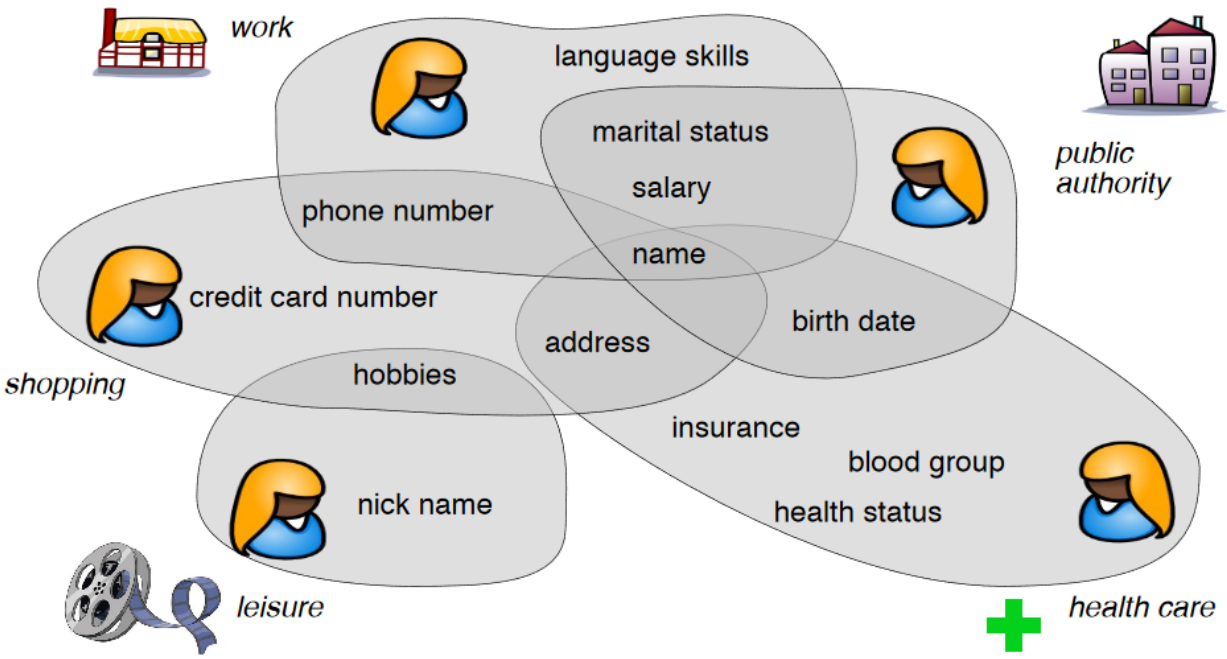
1985 ⟶ Today?



Even more connections...



Collecting all your data...(& IoT threat?)



Identity Theft & Collecting data

Breaches Related to Identity

Cost $15'000'000'000 worldwide (2015)

33% cyber crime take less than 5 min

ID worth: $> 100€$ in 2007 $\longrightarrow$ $< 1€$ today

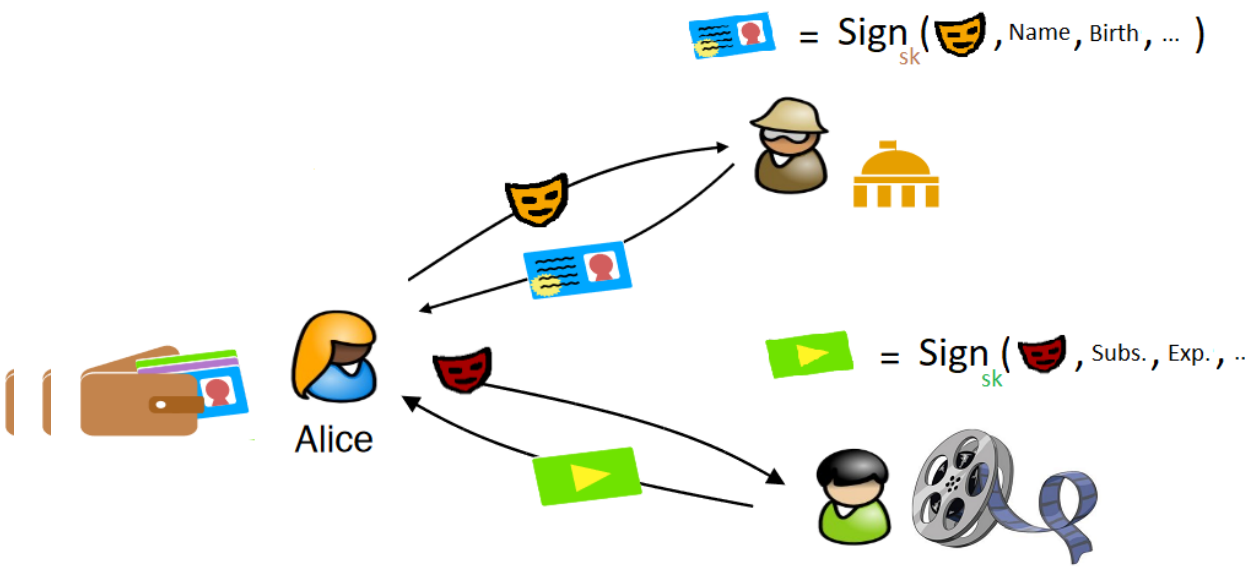
Learning from Snowden

NSA collects massive amounts of data

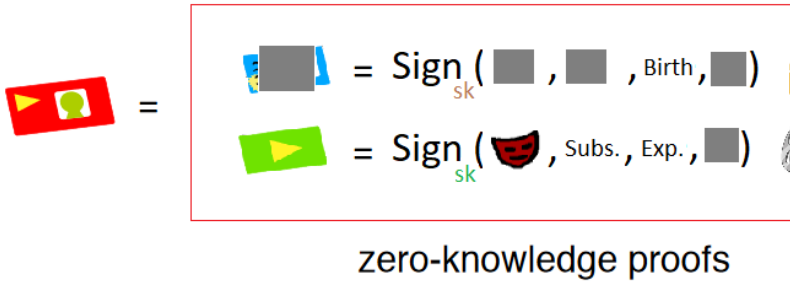
Not by breaking security! No cryptanalysis

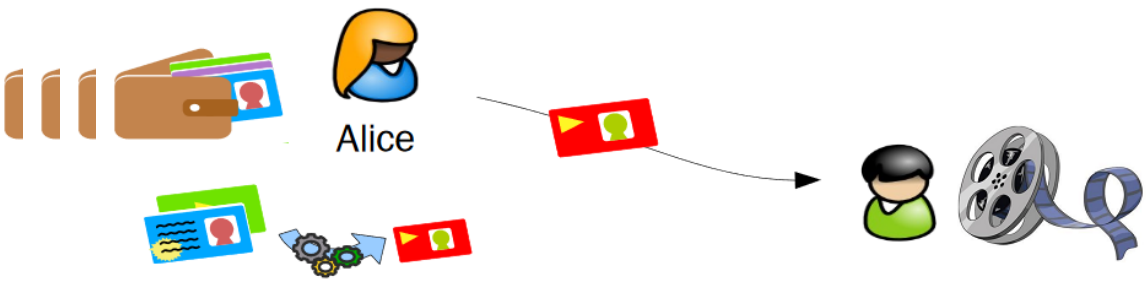
Jan Camenisch

Issuing Credentials

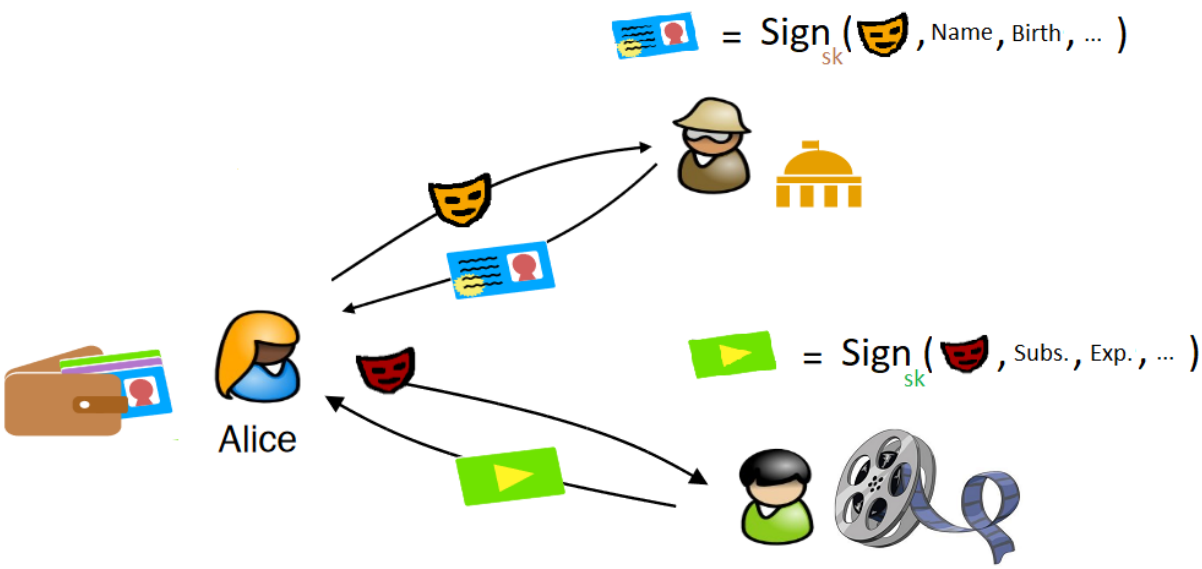


Accessing Services: Minimal Disclosure





Abuse: Mixing Cross-ID Credentials



Abuse: Mixing Cross-ID Credentials

Source of potential abuse

- Nothing links credentials and to Alice! What?
- Indeed, were and issued by a single user?
- Eavesdropping is stealing credential here

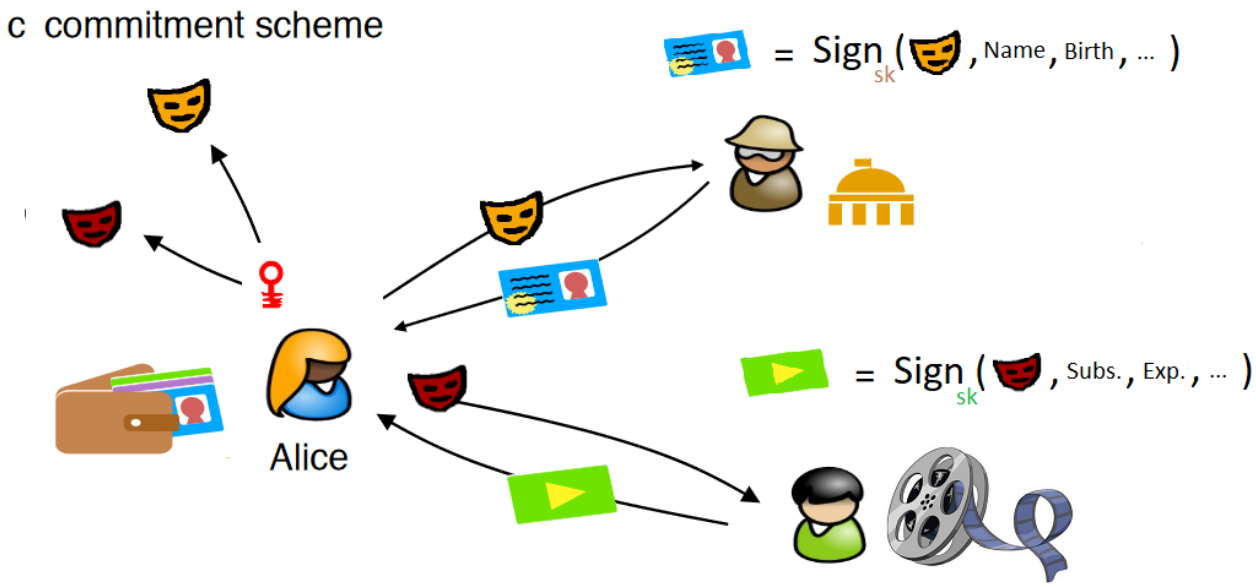
Attempt to prevent abuse

- Encrypt the credential? Still, what about malicious Alice...
- Remark: nothing comes from anonymity (e.g. login-password)

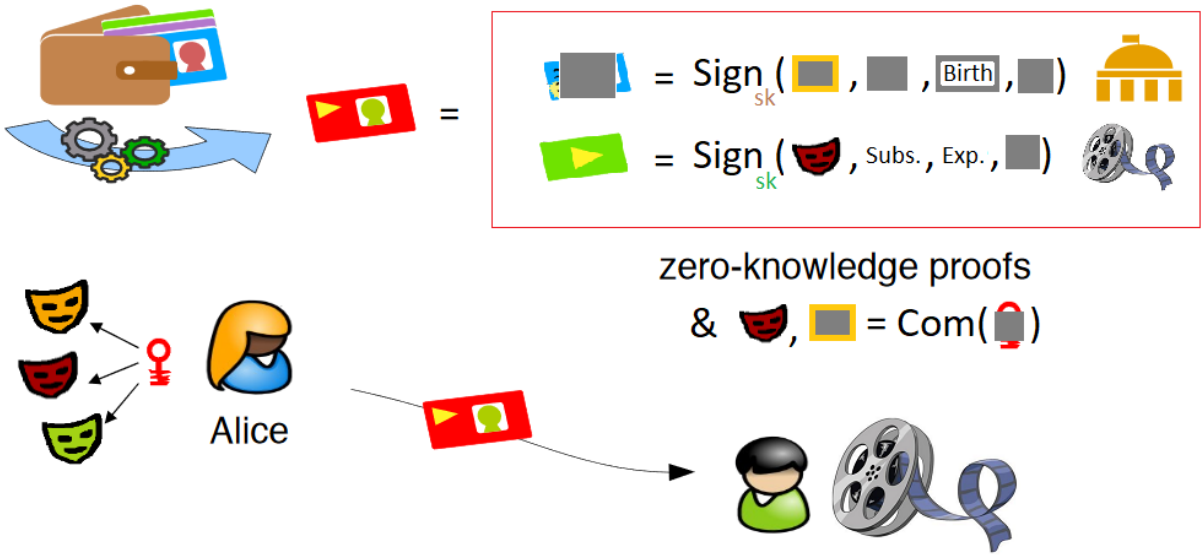
Better idea

- Prove there is a link . without revealing $\longrightarrow$ How?

Solution: Key Binding



Fair Service Access



Anonymous Credential

New pseudonym for each organization

- Several commitments  of a single secret key 

Issuing credential  from organization

- Prove validity of the commitment 
- Get "oblivious authentication" on  and attributes

Use credentials to show access rights

- Reveal some attributes, prove they are related to some 
- Minimal disclosure (ZKPK), hide  for anonymity

Anonymous Credential

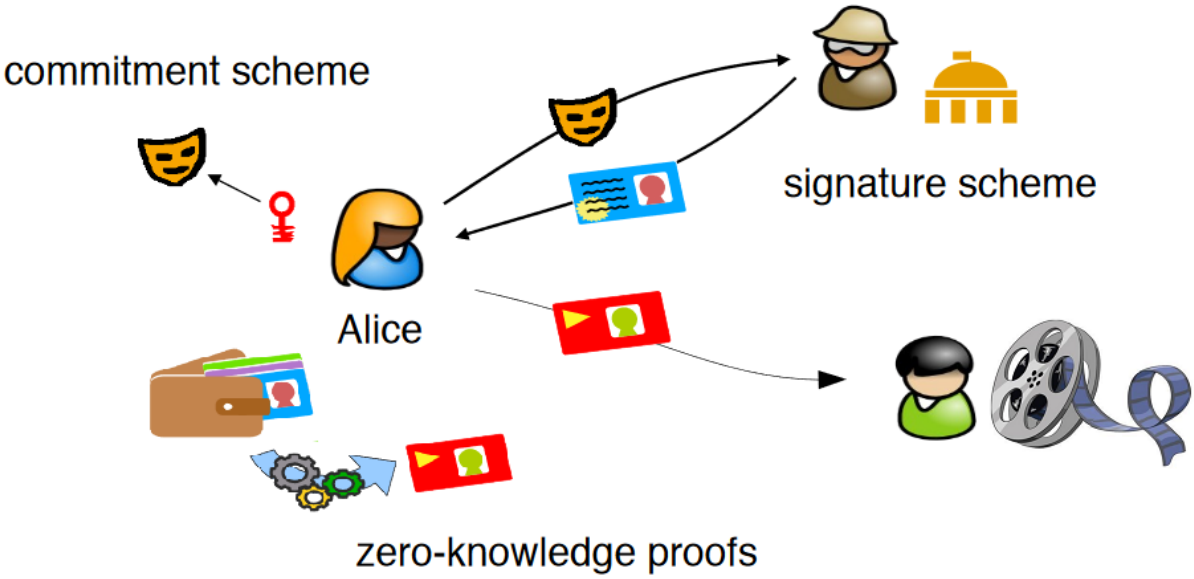
User privacy protection

- Unlikability (multi-use, several credentials from organizations)
- Combining multiple credentials, fined grained access
- Chosing what to disclose, proving predicate over attributes

Access right protection

- No possession of credential not owned ($\sim$ unforgeability)
- No cross-user combination of credentials ($\sim$ soundness)
- No wrong statement $\longrightarrow$ protecting organizations

Anonymous Credential



Building AC System

Generic solution

- Commitment, signature, zero-knowledge proof
- 2PC protocol $\longrightarrow$ theoretical efficiency, interactions?

Practical real-world scheme

- Idemix (IBM), U-Prove (Microsoft)

Additional features for practical use

- Revocation, accountability (encrypted identity?),...
- eCash: preventing double spending,...

Composable-Friendly Building Blocks

commitment scheme

signature scheme

zero-knowledge proofs

Composable-Friendly Building Blocks

Commitment scheme $c = \mathsf{Com}(\mathsf{usk})$

Think of $c =$  and $usk =$ 

Pedersen: $c = g^{\mathsf{usk}} h^r$ where $h \in \langle g \rangle = \mathbb{G}$

Pedersen Commitment: (recall, crypto background)

$\mathsf{Gen}(1^n)$: outputs $pk = (\mathbb{G}, q, g, h)$, n-bit prime order q

$\mathsf{Com}_{pk}(m; r)$: pick $r \leftarrow \mathbb{Z}_q$, compute $c = g^m h^r$ in $\mathbb{G}$

$\longrightarrow$ Perfectly hiding: c is random (independent of m)

$\longrightarrow$ Comput$^{\text{ly}}$ binding: cannot find other m', r' under DL

Issuing Credential from c

Sending $c =$  reveals nothing about $m = usk =$ 

Alice (ZK) proves she knows usk, r s.t. $\mathsf{Open}(c, m, r) = \top$

Alic needs a ignat e on , nd her ttribu e $a_1, \ldots, a_n$

Composable-Friendly Building Blocks

Commitment scheme $c = \mathsf{Com}(\mathsf{usk})$

Think of $c =$  $= g^{\mathsf{usk}} h^r$ and $\mathit{usk} =$ 

To be linked with attributes a_1 (age), a_2 (exp. date),...

Multi-Pedersen: (generalized commitment)

$\mathsf{Gen}(1^n)$: outputs $pk = (\mathbb{G}, q, g_1, \ldots, g_\ell, h)$, n-bit prime order q

$\mathsf{Com}_{pk}(m_1, \ldots, m_\ell; r)$: pick $r \leftarrow \mathbb{Z}_q$, output $c = g_1^{m_1} \cdots g_\ell^{m_\ell} h^r$

$\longrightarrow$ Perfectly hiding & Computationally binding (DL)

Another Pseudonym?

Randomizability: $c' = c \cdot h^{r'} = g_1^{m_1} \cdots g_\ell^{m_\ell} h^{r+r'} =$ 

$\longrightarrow$ Perfect unlinkability between  and 

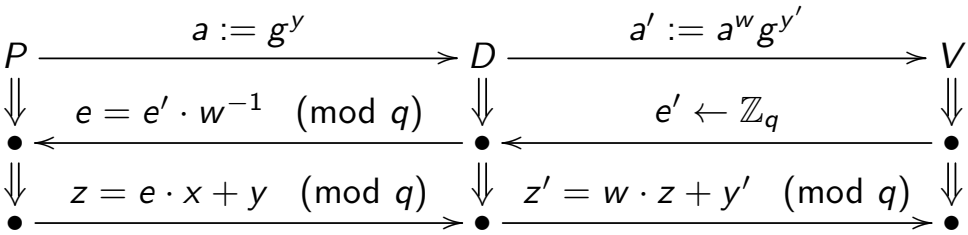
How to compute $\sigma = \mathsf{Sign}_{\mathsf{sk}}(c)$?

Issuing a Signature (1)

Divertible Schnorr Proof

P "allows" D to prove knowledge of its x to V who has $h := g^x$

Prover Verifier



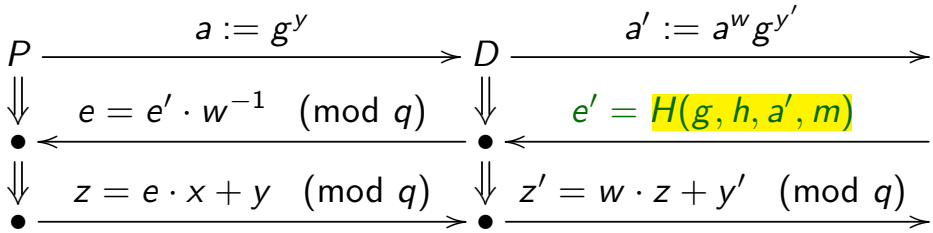
D verifies $h^e a = g^z$ V verifies $h^{e'} a' = g^{z'}$

Correctness: $g^{z'} = (g^z)^w g^{y'} = (h^e a)^w g^{y'} = h^{ew}(a^w g^{y'}) = h^{e'} a'$

Turn it into a Schnorr signature: D computes $e' = H(g, h, a', m)$

Issuing a Signature (2)

"Non-interactive" Divertible Schnorr's Proof



Anyone can verify that $e' = H(g, h, g^{z'}/h^{e'}, m)$ with $\sigma = (e', z')$ in $\mathbb{Z}_q^2$

Pro's: (a, e, z) and (a', e', z') are unlinkable

Con's: P has no control on the message m that he signed

Issuing a Signature (3)

Solution

- Let P see the message $c = u_1^{m_1} \dots u_\ell^{m_\ell} u_0^r$ and a proof π
- P computes $d = c^x$ and allows proving knowledge of

$$\log_g h = x = \log_c d$$

$\longrightarrow$ Divertible Chaum-Persersen made "non-interactive"

- Allows adapting (c, d) into randomized (c', d') so that the signature $\sigma = (e', z')$ is valid for

$$\log_g h = x = \log_{c'} d'$$

- Since $c' = c \cdot u_0^{r'}$, P gives $v_0 = u_0^x$ to help computing

$$d' = d \cdot v_0^{r'} = (c \cdot u_0^{r'})^x = (c')^x$$

Issuing a Signature (4)

u et v sont dans la clé publique
les u vont permettre de stocker les messages m_1 à m_l. u_0 c'est l'id du user, caché par l'exposant r dans le commit c

The final protocol: $\mathsf{pk} = (g, h, u_0, v_0, \{u_i\}_{i=1}^{\ell})$

Signer(x)		**User**$(m_1 = \text{🔑})$
Pick $y \leftarrow \mathbb{Z}_q$		Pick $r, r', w, y' \leftarrow \mathbb{Z}_q$
$d = c^x$	$\longleftarrow$ c and a proof π	$c = u_1^{m_1} \cdots u_\ell^{m_\ell} u_0^r$
$a = g^y, b = c^y,$ $a_0 = u_0^y$	$\longrightarrow$ $d,\ a,\ b,\ a_0$	$c' = c \cdot u_0^{r'},\ d' = d \cdot v_0^{r'}$ $a' = a^w g^{y'}$ $b' = (b \cdot a_0^{r'})^w (c \cdot u_0^{r'})^{y'}$
	$\longleftarrow$ $e = e' \cdot w^{-1}$	$e' = H(g, h, c', d', a', b')$
$z = ex + y \bmod q$	$\longrightarrow$ z	$z' = wz + y' \bmod q$

Issuing a 'Chaum-Pedersen'-like Signature

Alice subscribes to the streaming provider

Use secret key $m_1 = \mathsf{usk}$, and use pk of this organization
Pick a random $r \leftarrow \mathbb{Z}_p$ and compute $c_1 = u_1^{\mathsf{usk}} u_0{}^r \in \mathbb{G}$
Send c_1 along with a ZKPK π of the exponents (usk, r)

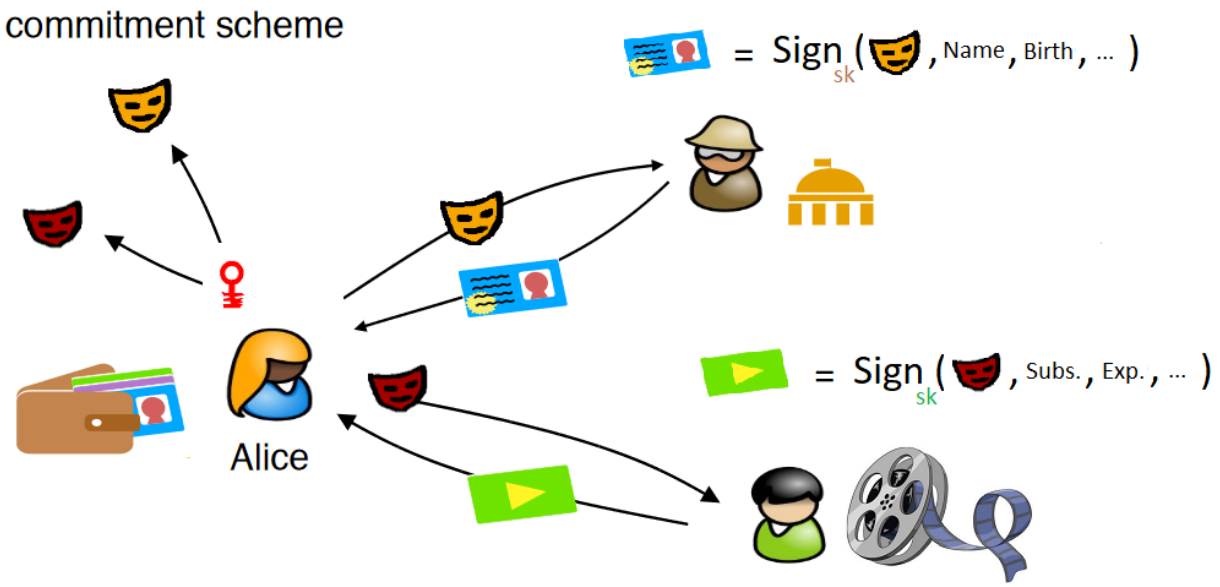
Streaming provider (blind) authentication

Check whether Alice paid and check the proof π for c_1
Add some attributes, say m_2 is the expiration date, $m_3 \ldots$
Compute $c = c_1 \cdot u_2^{m_2} \cdots u_\ell^{m_\ell} = u_1^{\mathsf{usk}} \cdot u_2^{m_2} \cdots u_\ell^{m_\ell} u_0^r$
Engage with Alice into a divertible Chaum-Pedersen NIZK-proof with $d = c^x$ so that Alice gets a signature $\sigma = (e', z')$ on a blinded message $(c', d') = (c, d) \cdot (u_0, v_0)^{r'}$ with $\log_g h = \log_{c'} d'$

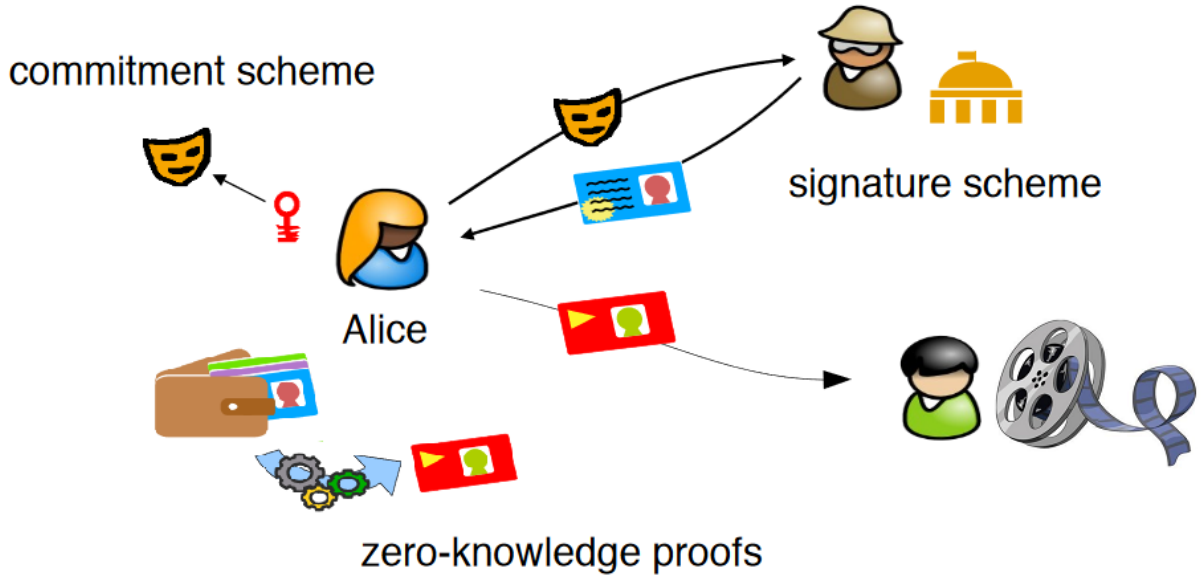
Alice stores her credential to watch movies

Verify the signature as $e' = H(g, h, c', d', g^{z'}/h^{e'}, (c')^{z'}/(d')^{e'})$
Store (c', d') and $\sigma = (e', z')$ for known $(\mathsf{usk}, m_2, \ldots, m_\ell; r + r')$

Issuing Credentials



Anonymous Credential



Privacy Enhancing Technologies

O. Pereira, T. Peters and F.-X. Standaert

Cryptocurrencies

eCash – 1983

BLIND SIGNATURES FOR UNTRACEABLE PAYMENTS

David Chaum

A fundamentally new kind of cryptography is proposed here, which allows an automated payments system with the following properties:

(1) Inability of third parties to determine payee, time or amount of payments made by an individual.

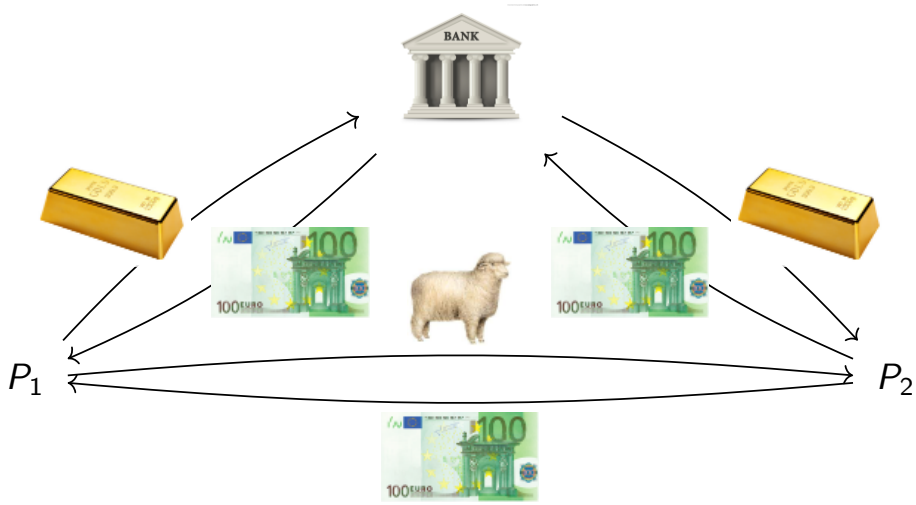
(2) Ability of individuals to provide proof of payment, or to determine the identity of the payee under exceptional circumstances.

(3) Ability to stop use of payments media reported stolen.

Commodity Money

- Social agreement on using a physical object as means of payment (goods, services, ...)
- Object chosen to be scarce, durable, transportable, ...
  $\Rightarrow$ metals look convenient

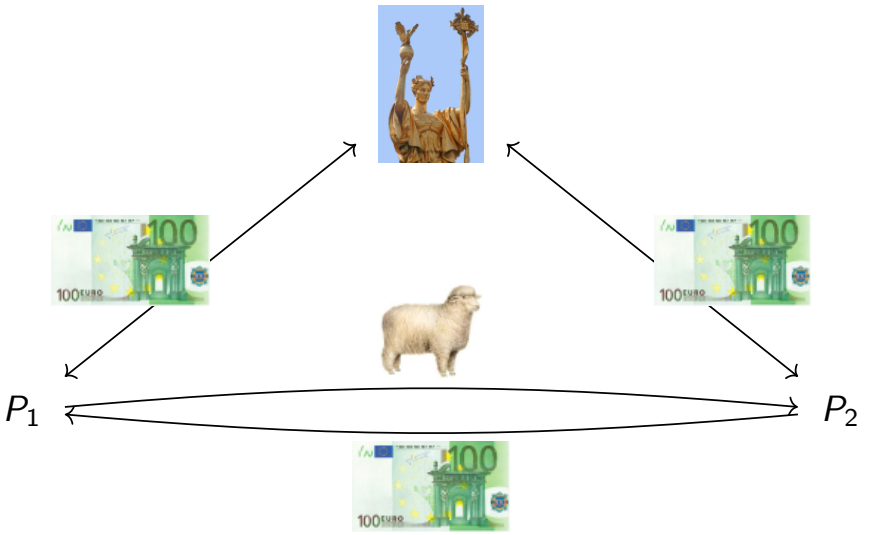
Representative Money



- Bank takes desposit, offers paper in exchange
- Paper offers much greater convenience, but easier to counterfeit

Fiat Money

actually



- Society decides value, no relation to any commodity
  $\Rightarrow$ no inherent limitation on creation, inter-society exchange ruled by mutual agreements

e-Cash

hard to copy

Goal:

- replace  with **2U3JO2891SJIS7389092**

Challenge:

- How to prevent double-spending?
  Digits can be perfectly replicated, for ever

e-Cash

Use online presence!

Solution 1:

- Give a unique *id* to each digital coin
- Bank used in each payment, to keep track of who owns each digital coin

But now bank can track all exchanges!

Cons : Not anonymous Bank
can cheat

Chaum – 1985

Payer:

- picks unique digital coin *id*
- *blinds* it, have it signed by bank, *unblinds* it
- pays by sending unblinded signed coin

Payee: Bénéficiaire

- verifies bank signature
- sends unblinded signed coin to bank
- waits until banks confirms that coin was not spent before
- validates transaction

Now bank cannot track exchanges anymore:

- bank only sees how much money each user has

Variant based on Anonymous Credentials

Using the Anonymous Credential scheme from the previous lecture:

$\text{KeyGen}(1^\lambda)$ Bank picks group $\mathbb{G}$ of primer order p with $|p| = \lambda$ generated by g
Pick random $sk = x \leftarrow \mathbb{Z}_p$ and set $h = g^x \in \mathbb{G}$
Pick verifiably random $(u_0, u_1) \in \mathbb{G}^2$ and set $v_0 = u_0^x$
Set $vk = (g, h, u_0, v_0, u_1)$, keep $sk = x$

$\text{NewCoin}(m)$ Payer picks $r \leftarrow \mathbb{Z}_p$, computes $c = u_1^m u_0^r$,
gets from Bank $\sigma = (e, z)$ on (c, d) where $d = c^x$
Bank's view is independent of the coin $((c, d), \sigma)$

$\text{Spend}(m, r, c, d, \sigma)$ Payer sends (m, r, c, d, σ) to payee
Payee checks that $c = u_1^m u_0^r$ and that σ is valid
Payee forwards (m, r, c, d, σ) to Bank, which also confirms whether m is fresh
Payee accepts transaction if all checks succeed

Offline e-Cash

Online bank is a big bottleneck!

But double-spending cannot be prevented without interaction:

- No way to detect if same coin spent with two different payees

Solution [Chaum, Fiat, Naor, 1988]:

- Forget prevention, focus on accountability ($\sim$ covert security)
- Make sure that any double-spending user will be caught

But how? Transactions are anonymous and unlinkable!

Making our scheme work offline

Idea:

- Bank generates more commitment bases u_2, u_3 as part of public key
- Each Payer U has secret key $sk_u \in \mathbb{Z}_p$ and public key $pk_u = u_2^{sk_u}$
- Each coin will contain sk_u as an attribute
- Modify **Spend** protocol so that:
  1. A coin spent only once leaks nothing about sk_u
  2. A coin spent twice reveals sk_U + id of cheater

  In practice:
  1. Also add a random t as an extra signed attribute
  2. Spending process includes Payee sending random R to Payer
  3. Payer must reveal $T = sk_u + Rt$ to payee

  When sk_u and t are secret, T reveals nothing about sk_u
  But $T_1 = sk_u + R_1 t$ and $T_2 = sk_u + R_2 t$ reveal sk_u and t

Making our scheme work offline

Adapting our previous scheme

$\text{KeyGen}(1^\lambda)$ Bank picks $\mathbb{G}, g, q$, x and $h = g^x$ as before
Picks $(u_0, u_1, u_2, u_3) \in \mathbb{G}^4$ and sets $v_0 = u_0^x$
Set $vk = (g, h, u_0, v_0, u_1, u_2, u_3)$, keeps $sk = x$

$\text{NewCoin}(m)$ Payer picks $r, r', t \leftarrow \mathbb{Z}_p^3$, computes $c' = u_1^m u_2^{sk_u} u_3^t u_0^{r+r'}$
proves to Bank that c' can be opened with sk_U on u_2
gets $\sigma = (e, z)$ on (c, d) where $c = c' u_0^{-r'}$ and $d = c^x$
Bank's view is independent of the coin $((c, d), \sigma)$

$\text{Spend}(m, r, c, d, \sigma)$ Payer sends (m, c, d, σ) to payee
with a proof that it can open c with m on u_1
Payee verifies proof and validity of σ
Payee sends $R \leftarrow \mathbb{Z}_p$ to Payer
Payer replies with $T = sk_U + Rt$
with proof that T is consistent with c and R

Payee keeps everything for further verification with Bank
If U spends the same coin twice, Bank can produce sk_u

Chaumian e-Cash

We can make digital coins s.t.:

- payers cannot be traced: parties, amounts are hidden
- payments can be proven, or double-spending parties are caught
- stolen coins can be reported and canceled before they are spent, or can be traced after spending

Numerous improvements:

- make coins divisible
- make protocols more efficient
- variants for specific uses: pre-paid access cards, ...
- ...

Remaining constraint:

- Need of a signing authority

Chaumian e-Cash

Adoption:

- Chaum created Digicash in 1989, bankruptcy in 1998
  Solution adopted by several banks in the 1990's
- Remains active research topic including at Orange, IBM, ...

Challenges:

- Patents created adversarial reaction from community
- Asymmetry between payers and payees can create difficulties
- Convenience of credit cards prevailed

Bitcoin

Introduction to the bitcoin protocol and some of the questions that it raises:

- Tutorial by Stefan Dziembowski
  (link on last slide + Moodle)

Anonymous cryptocurrencies

Bitcoin:

- pseudonymous-backed payments
- but fully distributed and symmetric architecture

Chaumian e-cash:

- anonymous payments
- but trusted authority needed to sign and redeem coins

Would it be possible to make *real* anonymous payments?

Zerocoin

An example of anonymous cryptocurrency:

- Protocol proposed by Miers, Garman, Green, Rubin – 2013
- Base of Zcash
  launched Oct. 2016, market cap: $\approx$ 500M in Oct. 2024

Zerocoin

Could we just put Chaumian e-cash on a blockchain?

- No signature authority available

$\Rightarrow$ Use blockchain as a public list of signed documents
"old-school digital signatures"

Bolt [2016]

Blind Off-chain Lightweight Transactions

- Push most of the transactions off-ledger
- Ledger used to open/close payment channels and dispute resolution

Basic idea:

- Merchants play role of bank in Chaumian e-cash
- User creates payment channel by zerocash-style deposit to merchant
  User engages in e-cash coin issuing for the equivalent amount
- User spends its e-cash

Properties:

- Cost of zerocash only comes on initial deposit
- Merchant does not know who is spending
- Merchant cannot link payments between channel users
- User can open unspent coins to close payment channel

Conclusion

Numerous attempts until Bitcoin deployment breakthrough

Numerous research questions raised:

- Technical:
  - Refresh of 1980-1990 technologies on which Bitcoin is based
  - Renewed interest and new research lines:
    - What are the properties of these new protocols?
    - Design of new distributed consensus algorithms
- Social:
  - What fosters user adoption?
  - How are cryptocurrencies managed, updated, . . . ?
- Economical/political:
  - What will be the effect/role of cryptocurrencies?
  - How do “old” and “new” currencies interact?

Privacy Enhancing Technologies

O. Pereira, T. Peters and F.-X. Standaert

Secure Computation

Secure Computation – 1982

Protocols for Secure Computations
(extended abstract)

Andrew Yao

Suppose m people wish to compute the value of a function $f(x_1, x_2, x_3, ..., x_m)$, which is an integer-valued function of m integer variables x_i of bounded range. Assume initially person P_i knows the value of x_i and no other x's. Is it possible for them to compute the value of f, by communicating among themselves, without unduly giving away any information about the values of their own variables? The millionaires' problem corresponds to the case when $m = 2$ and $f(x_1, x_2) = 1$ if $x_1 < x_2$, and 0 otherwise.

Secure Computation

The most general distributed computing problem!

Secure function evaluation:

- Each party $P_i \in \{P_1, \ldots, P_n\}$ has a private input x_i
- Each P_i is expected to learn $f_i(x_1, \ldots, x_n)$ but nothing else

Examples:

- Secure communication from P_1 to P_2, with adversary P_3:
  $f_1(x_1, \perp, \perp) = \perp$, $f_2(x_1, \perp, \perp) = x_1$, $f_3(x_1, \perp, \perp) = |x_1|$
- Authentic communication from P_1 to P_2, with adv. P_3:
  $f_1(x_1, \perp, \perp) = \perp$, $f_2(x_1, \perp, \perp) = x_1$, $f_3(x_1, \perp, \perp) = x_1$

Secure Computation

More examples:

- Election: $f_i(x_1, \ldots, x_n) := \sum_{i=1}^{n} x_i \geq^? \frac{n}{2}$
- Vickrey Auction: $f_i(x_1, \ldots, x_n) := (j, \max(\{x_1, \ldots, x_n\} - \{x_j\}))$,
  where $j := \text{argmax}(\{x_1, \ldots, x_n\})$
- Border control:
  $f_1(x_1, x_2) := \perp$, $f_2(x_1, x_2) := (x_2 \in^? x_1)$
  where x_1 is a database of suspect faces,
  x_2 is a face seen at border
- Trading list of phishing website URLs:
  $f_1(x_1, x_2) = |x_2 \setminus x_1|$, $f_2(x_1, x_2) = |x_1 \setminus x_2|$
  where x_i is a set of malicious URLs known by P_i

Secure Computation

More examples:

- Secure distributed signature with verification key for $sk_1 \oplus sk_2$:
  $f_1(m, sk_1, sk_2) = Sign_{sk_1 \oplus sk_2}(m)$,
  $f_2(m, sk_1, sk_2) = f_3(m, sk_1, sk_2) = m$

Most of these applications were already envisioned by Yao!

> These results have applications to secret voting, private querying of database, oblivious negotiation, playing mental poker, etc. .

Secure Computation

Continous research since Yao in early 1980's

Actual products and companies:

- Auctions in Denmark – Partisia
- Voting in Belgium – Bluekrypt
- Database for private computation in Estonia – Cybernetica
- Distributed security module in Israel – Unbound Tech (now Coinbase)
- Uses by Google, Microsoft, SAP, ...
- ...

Secure Computation

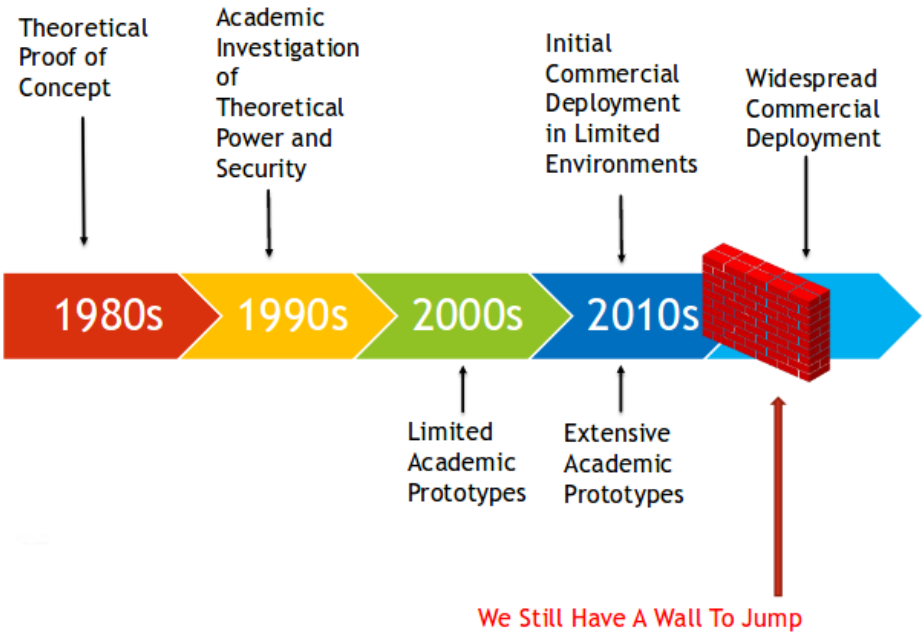


Image credit: N. Smart

Secure Computation – What can we do?

General feasibility [GMW86, CCD88, BGW88]:

Secure evaluation of any boolean circuit!

Numerous settings and restrictions:

- 2/3 majority of honest players with point-to-point communication
- simple majority of honest players with broadcast channel
- any level of corruption with broadcast channel
  but computational security *only* and no guarantee of output delivery

Secure Computation – What can we do?

Efficiency:

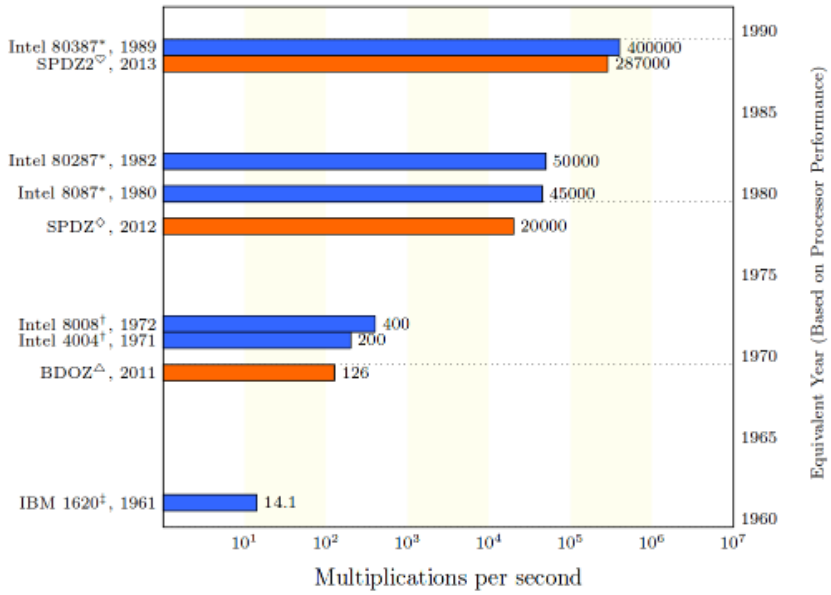


Image credit: P. Scholl

Secure Computation – Why don't we do it more?

Main practical issues:

- Speed: computational power, bandwidth, latency due to round complexity
- Practical difficulty of having multiple parties running a complex protocol together and independently
- Many functions do not have a compact representation as a circuit (data dependent branching, memory access, . . .)

Currently restricts adoption to specific applications: static and simple enough function and/or few time constraints

But that makes a lot of useful functions already!

Two-Party Computation

Our focus in this class:

Two-Party Computation

Motivation:

- Fastest: complexity often quadratic in # of parties
- Easier deployment: only two parties need to participate
- Sufficient for many applications
- Captures many of the techniques used in the general case

Two parties $\Rightarrow$ computational security, no output guarantee

Two-Party Computation – Definition

What protocol would we like, in an *ideal world*?

- P_1 and P_2 secretly hand x_1 and x_2 to a trusted party $\mathcal{F}$
- $\mathcal{F}$ returns:
  - $f_1(x_1, x_2)$ to P_1 and
  - $f_2(x_1, x_2)$ to P_2

Can we hope to be that good in the *real world*, without $\mathcal{F}$?

- Too hard in the 2-party case:
  one party learns its output first, and may just abort the protocol before the other party does

Two-Party Computation – Adversarial Power

Interesting case:

- one party is corrupted, i.e., controlled by adversary $\mathcal{A}$

Two honest parties make the problem trivial:

- One of the parties can play the role of $\mathcal{F}$

Two corrupted parties make the problem trivial:

- If noone follows the rules, no guarantee can be given to anyone

One corrupted party:

- What guarantees can the protocol give to a honest party if the other party is corrupted?

Adversarial Power – Some Vocabulary

Various deviation types:

- *Semi-honest* or *honest-but-curious*: $\mathcal{A}$ follows the protocol, but tries to learn what he can
  Can monitor the party device, not modify it.
- *Malicious*: corrupted party behaves arbitrarily
- *Covert*: corrupted party can deviate, as long as his deviations cannot be detected

Various corruption types:

- *Static*: corruption happens before the protocol starts
- *Dynamic*: corruption happens at any time

Our focus: semi-honest $\mathcal{A}$ and static corruption

Two-Party Computation – Definition

What protocol would we still like, in an *ideal world*?

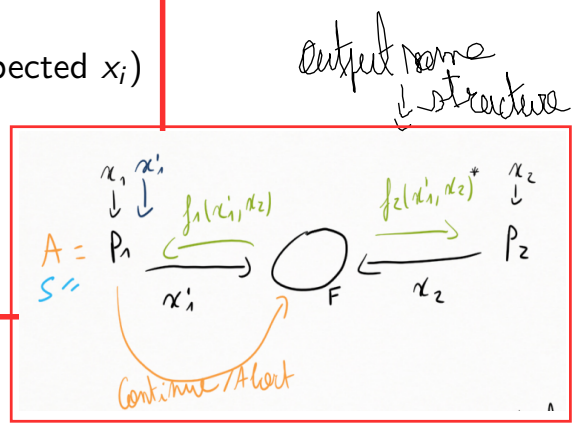
Let:

- $\mathcal{F}$ be a trusted party, who knows who is corrupted
- P_i be corrupted – P_j be honest

$\mathcal{F}$ behaves as follows:

- receive x_i' for P_i (x_i' may not be the expected x_i)
- receive x_j from P_j
- return $f_i(x_i', x_j)$ (or $f_i(x_j, x_i')$) to P_i
- if P_i answers "abort", then halt
- else return $f_j(x_i', x_j)$ (or $f_j(x_j, x_i')$) to P_j

Now, P_i controls delivery of output to P_j



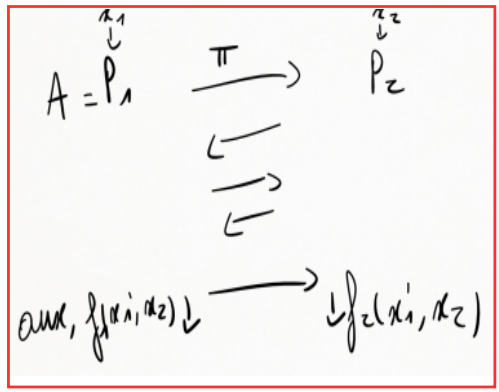
Two-Party Computation – Definition

We now would like:

- a *real-world* protocol π between P_1 and P_2 (no $\mathcal{F}$)
- such that P_i cannot do anything that cannot be done in the *ideal world*

This idea is the core of the:

ideal-world/real-world
definition paradigm



Two-Party Computation – Definition

Let n be the security parameter and

- $\text{REAL}_{\pi,\mathcal{A},i}(x_1, x_2, n)$ be:
  - the output of $\mathcal{A} = P_i$ when running π when (P_1, P_2) have inputs (x_1, x_2)
  - together with the output of P_j
- $\text{IDEAL}_{\mathcal{F},\mathcal{A},i}(x_1, x_2, n)$ be:
  - the output of $\mathcal{A} = P_i$ when interacting with $\mathcal{F}$ when (P_1, P_2) have inputs (x_1, x_2)
  - together with the output of P_j

Protocol π *securely emulates* $\mathcal{F}$ if,
$\forall \mathcal{A}$ against π, $\exists \mathcal{S}$ against $\mathcal{F}$, such that:

$$\text{REAL}_{\pi,\mathcal{A},i}(x_1, x_2, n) \approx \text{IDEAL}_{\mathcal{F},\mathcal{S},i}(x_1, x_2, n)$$

for $i \in \{1, 2\}$, $x_1, x_2 \in \{0, 1\}^*$, $|x_1| = |x_2|$

Protocol π securely emulates $\mathcal{F}$

What does P_j get when playing π?

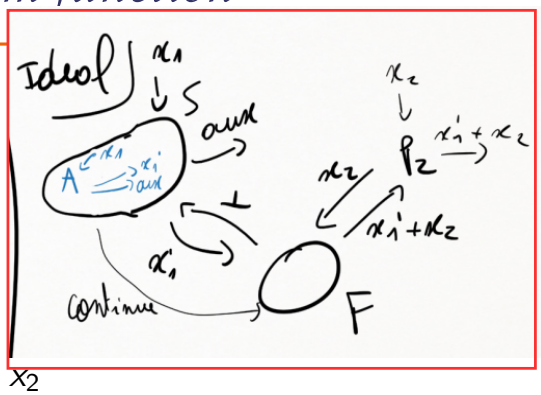
Privacy P_i does not learn anything about x_j, except for what can be deduced from f_i. Otherwise, $\mathcal{S}$ could not produce indistinguishable output in ideal world.

Correctness P_j's output follows $f_j(x_i', x_j)$ (or $f_j(x_j, x_i')$) for an x_i' given by P_i

Independence P_i cannot choose x_i' as a function of x_j (Think: auction, coin flipping game, ...)

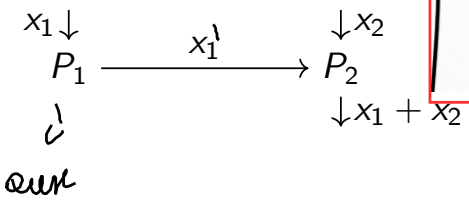
No fairness P_i may learn f_i while P_j learns nothing

Example 1: the single-sided sum function



Let $f_1(x_1, x_2) = \perp$ and $f_2(x_1, x_2) = x_1 + x_2$

Let π be as follows:

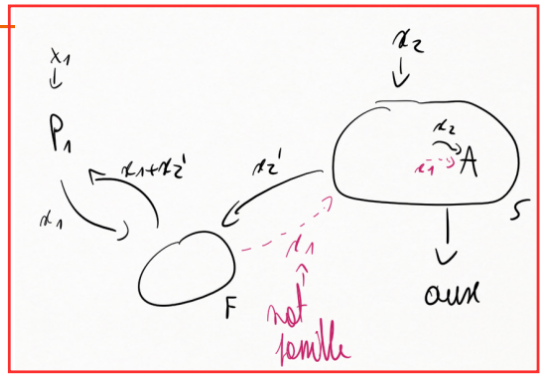


Does π securely emulate $\mathcal{F}(f_1, f_2)$?

1. Can we build a simulator when P_1 is corrupted?
2. Can we build a simulator when P_2 is corrupted?

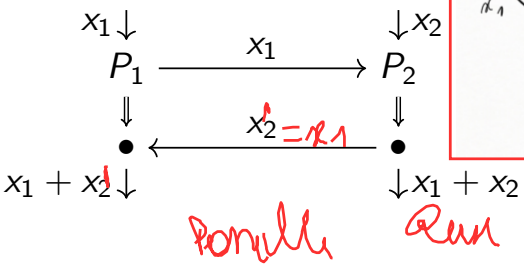
A distribute in the same way so "aux" is distributed in the same way

Example 2: the two-sided sum function



Let $f_1(x_1, x_2) = f_2(x_1, x_2) = x_1 + x_2$

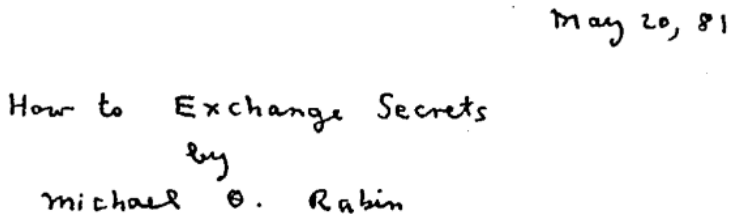
Let π be as follows:



Does π securely emulate $\mathcal{F}(f_1, f_2)$?

1. Can we build a simulator when P_1 is corrupted?
2. Can we build a simulator when P_2 is corrupted?

"inconscient" *Oblivious Transfer (OT)*



Michael Rabin
Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.

Oblivious Transfer (OT)

b = secret bit = le message reçu avec m

Oblivious transfer (1-2):

$$f_1((m_0, m_1), b) = \perp,\ f_2((m_0, m_1), b) = m_b$$

- P_1 is called the sender, P_2 the receiver
- P_2 learns 1 out of 2 messages, P_1 does not know which one
- Proposed for contract signing
- Simple private information retrieval/oblivious memory: Make 1-n Oblivious Transfer on n stored files
- Complete functionality for secure computation [Kilian'88]

OT based on homomorphic encryption

Additively homomorphic encryption:

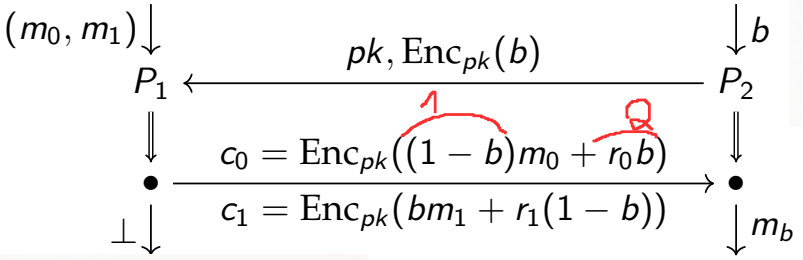
- $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$, $c \leftarrow \mathrm{Enc}_{pk}(m)$, $m = \mathrm{Dec}_{sk}(c)$
- If $c_1 \in \mathrm{Enc}_{pk}(m_1)$ and $c_2 \in \mathrm{Enc}_{pk}(m_2)$, then $c_1 \cdot c_2 \in \mathrm{Enc}_{pk}(m_1 + m_2)$ and $c_1^a \in \mathrm{Enc}_{pk}(a \cdot m_1)$

Example: Exponential ElGamal

- $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$ with:
  - $sk = x$ with $x \leftarrow \mathbb{Z}_q$ and $|q| = n$
  - $pk = (\mathbb{G}, g, q, h)$ with $\mathbb{G}$ a group of prime order q generated by g and $h = g^x$ with $\mathcal{M} = \mathbb{Z}_q$
- $\mathrm{Enc}_{pk}(m) = (g^r, g^m h^r)$ with $r \leftarrow \mathbb{Z}_q$
- $\mathrm{Dec}_{sk}(c_1, c_2) = \log_g(c_2/c_1^x)$ – works if m is small enough
- Secure if DDH is hard in $\mathbb{G}$, i.e., $(g, g^x, g^y, g^{xy}) \approx (g, g^x, g^y, g^z)$ for random $x, y, z \leftarrow \mathbb{Z}_q$

OT based on homomorphic encryption

Passively secure OT protocol π_{OT}:



Here:

- r_0, r_1 are random such that $m_0, m_1, r_0, r_1 \in \{0,1\}^\ell \subset \mathcal{M}$
- Do you see how P_1 computes c_0 and c_1?
- $m_b = \mathrm{Dec}_{sk}(c_b)$ and $r_{1-b} = \mathrm{Dec}_{sk}(c_{1-b})$

Secure in honest but curious

OT based on homomorphic encryption

Passive/honest-but-curious security:

- If P_1 and P_2 follow π_{OT}, then π_{OT} securely emulates $\mathcal{F}_{OT}$

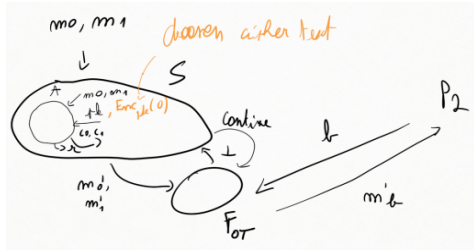
In particular:

- P_1 learns nothing about b
- P_2 learns nothing about m_{1-b} but learns m_b
- (b cannot be chosen as a function of the m_i and the m_i cannot be chosen as a function of b)

π_{OT} security: P_1 corrupted

$\mathcal{S}_1$ behaves as follows, on input (m_0, m_1):

- Start an instance of $\mathcal{A}_1$ on (m_0, m_1) and get (m'_0, m'_1)
- Submit (m'_0, m'_1) to $\mathcal{F}_{OT}$ and get back $\perp$
- Run $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$ and compute $c \leftarrow \mathrm{Enc}_{pk}(0)$
- Compute (c_0, c_1) as a honest P_1 would do from both (m'_0, m'_1) and P_2's message (pk, c)
- Send "continue" to $\mathcal{F}_{OT}$
- Set view as $(m'_0, m'_1, pk, r_0, r_1, c_0, c_1)$



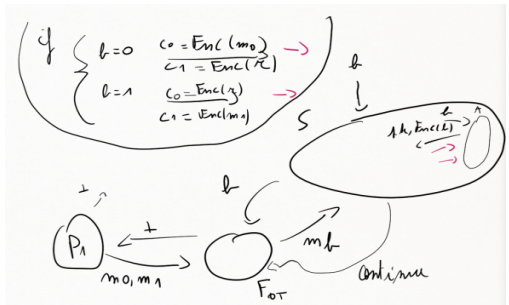
Analysis:

- P_2 gets correct output: guaranteed by $\mathcal{F}_{OT}$
- $\mathcal{S}_1$'s output is indistinguishable from $\mathcal{A}_1$'s real view: If $b = 0$, it is perfectly indistinguishable, and, if $b = 1$ it is not feasible to distinguish $\mathrm{Enc}_{pk}(1)$ from $\mathrm{Enc}_{pk}(0)$

π_{OT} security: P_2 corrupted

$\mathcal{S}_2$ behaves as follows, on input b:

- Start an instance of $\mathcal{A}_2$, give it b and gets $b' \in \{0,1\}$
- Run $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$ and compute $c \leftarrow \mathrm{Enc}_{pk}(b')$
- Send b' to $\mathcal{F}_{OT}$ and get back $m_{b'}$
- Send "continue" to $\mathcal{F}_{OT}$
- Compute c_0, c_1 as follows: pick random $r \leftarrow \{0,1\}^\ell$
  - $\mathrm{Enc}_{pk}(m_{b'}), \mathrm{Enc}_{pk}(r)$ if $b = 0$
  - $\mathrm{Enc}_{pk}(r), \mathrm{Enc}_{pk}(m_{b'})$ if $b = 1$
- Set view as $(b', pk, sk, c, c_0, c_1, m_{b'})$



Analysis:

- P_1 gets correct output (i.e. $\perp$)
- The distribution of $\mathcal{S}_2$'s output is identical to the distribution of $\mathcal{A}_2$'s view in real world

π_{OT} security

1. Can a malicious P_1 cheat, by behaving arbitrarily?
   Yes. E.g.: Make "normal" c_0 and impossible to decrypt c_1.
   P_2 will produce an output only if $b = 0$
2. Can a malicious P_2 cheat, by behaving arbitrarily?
   Yes. E.g.: Let $m_0, m_1 \in \{0,1\}$, $\ell = 1$, and send $(pk, \mathrm{Enc}_{pk}(2))$
   Then $\mathrm{Dec}_{sk}(c_0)$ and $\mathrm{Dec}_{sk}(c_1)$ leak both m_0 and m_1.

Various solutions are possible.
E.g., add ZK proofs.

Yao's Garbled Circuits

Yao's solution to the 2-party computation problem:

- Honest-but-curious parties
  Highly non-trivial already!

Oblivious Transfer

- Uses OT and Symmetric Encryption: much more efficient
- Focuses on the case $f_1(x_1, x_2) = \perp$
  But not a problem: given general (f_1, f_2), evaluate:
  - $f_1'((x_1, r), x_2) = \perp$ for a random r and
  - $f_2'((x_1, r), x_2) = (f_1(x_1, x_2) \oplus r, f_2(x_1, x_2))$

  and ask P_2 to send $f_1(x_1, x_2) \oplus r$ back to P_1 in the end.

There is a Generator and a Evaluator

Yao's Garbled Circuits

P1 compute le circuit tronqué ET P1 compute la Truth table version crypto.
Il envoie le circuit et la colonne des outputs encrypté.
(attention a shuffle cette colonne sinon on peut deviner)

Strategy:

1. Express f_2 as a boolean circuit, with $2 - 1$ gates
2. P_1 computes a *garbled* version of f_2 and sends it to P_2
   This is independent of (x_1, x_2)
3. P_1 sends the "ungarbling" keys associated to x_1
4. P_1 obliviously transfers the "ungarbling" keys associated to x_2
5. P_2 evaluates the circuit thanks to these keys

Garbling a Circuit

Strategy:

1. **Obfuscation** For each wire w of the circuit, pick random k_w^0 and k_w^1
   But for output wires: set $k_w^0 = 0$ and $k_w^1 = 1$
2. **Gate connection** For each gate g with inputs wires a, b and output wire c, compute table:

$$\begin{array}{ll} \mathrm{Enc}_{k_a^0}(\mathrm{Enc}_{k_b^0}(k_c^{g(0,0)} \| 0^n)) & \mathrm{Enc}_{k_a^0}(\mathrm{Enc}_{k_b^1}(k_c^{g(0,1)} \| 0^n)) \\ \mathrm{Enc}_{k_a^1}(\mathrm{Enc}_{k_b^0}(k_c^{g(1,0)} \| 0^n)) & \mathrm{Enc}_{k_a^1}(\mathrm{Enc}_{k_b^1}(k_c^{g(1,1)} \| 0^n)) \end{array}$$

   Enc and 0^n chosen such that Dec with wrong keys is visible, but no way to see which key(s) is/are wrong
3. **Shuffle** Shuffle the content of each table

Observe:

- Given keys associated to input wires, I can "ungarble" the whole circuit, but do not learn anything about the value on those wires.

Transmitting Inputs

Strategy:

1. P_1 knows keys associated to his inputs:
   Just send these keys to P_2
2. P_1 does not know P_2's keys
   OT of (k_a^0, k_a^1), and P_2 picks key associated to his input bit

Yao's Garbled Circuits - P_1 corrupted

$\mathcal{S}_1$ behaves as follows, on input x_1:

- Start an instance of $\mathcal{A}_1$, give it x_1 (and get $x_1'\ldots$)
- Create garbled circuit and keys associated to x_1 (as $\mathcal{A}_1$ would do)
- Submit x_1 to $\mathcal{F}_{f_2}$
- When receiving $\perp$ from $\mathcal{F}_{f_2}$: Run the OT simulator (corrupted P_1) with $\mathcal{A}_1$ for each of the keys associated to x_2
- Send "continue" to $\mathcal{F}_{f_2}$
- Set the view in the appropriate order

Observe:

- P_2's output is unchanged,
- $\mathcal{A}_1$'s view is unchanged, up to the quality of OT simulation

Yao's Garbled Circuits - P_2 corrupted

$\mathcal{S}_2$ behaves as follows, on input x_2:

- Give x_2 to an instance of $\mathcal{A}_2$, (get $x_2'\ldots,$) and to $\mathcal{F}_{f_2}$
- Receive $z = f_2(x_1, x_2)$ from $\mathcal{F}_{f_2}$ (when P_1 submits x_1)
- Create a "fake" garbled circuit as follows ($\mathcal{S}_2$ ignores x_1):
  - Pick random k_w^0 and k_w^1 for each wire w
  - For each gate with inputs a, b and output c, compute table:

$$\begin{array}{cc} \mathrm{Enc}_{k_a^0}(\mathrm{Enc}_{k_b^0}(k_c^0\|0^n)) & \mathrm{Enc}_{k_a^0}(\mathrm{Enc}_{k_b^1}(k_c^0\|0^n)) \\ \mathrm{Enc}_{k_a^1}(\mathrm{Enc}_{k_b^0}(k_c^0\|0^n)) & \mathrm{Enc}_{k_a^1}(\mathrm{Enc}_{k_b^1}(k_c^0\|0^n)) \end{array}$$

    And for each output gate, use z_i instead of k_c^0
    $\Rightarrow$ for all input keys, evaluation gives the right output z
  - Shuffle tables
- Send k^0 associated to each of $P_1's$ input wire
  $\Rightarrow$ ciphertexts are shuffled, this does not matter
- Run the OT simulator (corrupted P_2) for each OT, with x_{2i} and the associated wire k^0 as output.
- Send "continue" to $\mathcal{F}_{f_2}$ and output the ordered view

Yao's Garbled Circuits - P_2 corrupted

Why does it work?

- P_1's view is unchanged
- P_2's view has the following changes:
  - Garbled circuit encrypts same key for all inputs
    But $\mathcal{A}_2$ only sees 2 input keys, and so cannot detect that change thanks to the security of the encryption scheme
  - Keys related to "x_1" are uniform and in a random order w.r.t. shuffled ciphertexts
  - Simulated OT instead of real OT
    But this is indistinguishable thanks to the quality of the OT simulation and the fact that the recieved key is always uniform

Yao's Garbled Circuits - Efficiency

Round complexity: 2-3 rounds can work!

- P_2 sends 1st OT message
- P_1 sends garbled circuit, keys for his inputs and OT 2nd message
- P_2 sends his output back to P_1 (if needed)

Computational complexity:

- $|x_2|$ OT's: $\approx 10|x_2|$ exponentiations for P_1 and
  $\approx 3|x_2|$ exp. for P_2 + $|x_2|$ DLog for P_2 (or ROM)
- Garbled circuit for c gates: $8c$ Enc for P_1, $2c$ Dec for P_2

Communication complexity:

- P_2 sends $\approx 2|x_2|$ elements of $\mathbb{G}$ and possibly $|f_2(x_1, x_2)|$ bits
- P_1 sends $4c$ ciphertexts, $|x_1|$ keys and $4|x_2|$ elements of $\mathbb{G}$

What breaks with malicious adversaries?

- Our OT protocol is insecure, as seen before
- P_1 can garble a different circuit, making sure that he gets more outputs in the end
- P_2 can change the output that it sends to P_1 in the final step

Improvement: The Free XOR trick

Two ideas:

- For each wire w, P_2 sees either k_w^0 or k_w^1 but never both
  $\Rightarrow$ Pick a global random secret $R \neq 0^n$ and set $k_w^1 = k_w^0 \oplus R$, for all wires
- For each XOR gate with inputs (k_a^0, k_a^1) and (k_b^0, k_b^1), define:
  - output $k_c^0 = k_a^0 \oplus k_b^0 = k_a^1 \oplus k_b^1$ and
  - output $k_c^1 = k_a^0 \oplus k_b^1 = k_a^1 \oplus k_b^0 = k_c^0 \oplus R$

  When evaluation happens, $k_a^i \oplus k_b^j = k_c^{i \oplus j}$

Result:

- Less randomness is needed
- XOR gates become "free": no need to encrypt/decrypt, no communication

[Kolesnikov, Schneider, ICALP'08]

Conclusion

We showed how to perform 2-party computation securely with:

- honest-but-curious adversaries and static corruption

Dozens of variants exist:

- numerous other efficiency improvements
- stronger adversarial models (covert, malicious, . . .)
- multiple parties
- information theoretic security
- . . .

A very useful tool when multiple distrustful parties need to work together!

Privacy Enhancing Technologies

O. Pereira and F.-X. Standaert

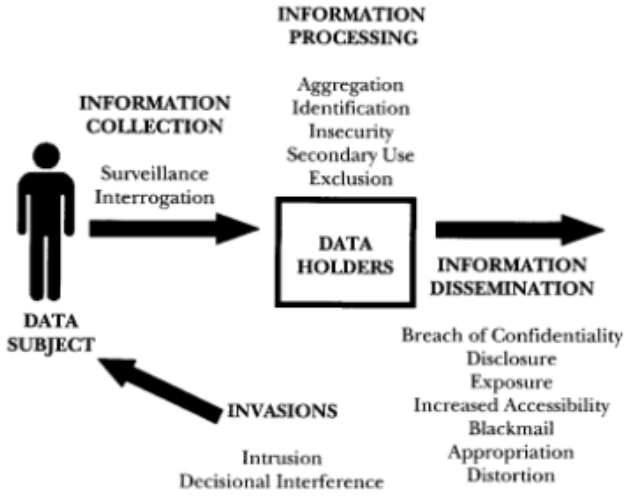
UCL Crypto Group, Université catholique de Louvain

Big Data Privacy (2 lectures)

Outline

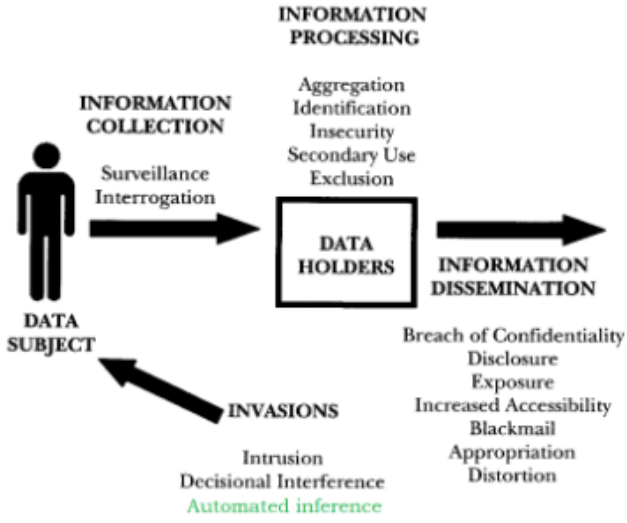
- Introduction (foreword, motivation and goal)
- Legal taxonomy and (partial) solutions
- Model 1: open (sanitized) data
  - Pseudonyms and limitations
  - *k*-anonymity, extensions and limitations
  - IT metrics and limitations
  - Extension: from Orwell to Kafka (& utility)
- Model 2: trusted DB with sanitized answers
  - Differential privacy and limitations
- Model 3: Untrusted DB (MPC, FHE and limitations)

Legal taxonomy (Solove 2006)



Technical solutions' initial (main) focus: identification (≈ Orwell's methaphor)

Legal taxonomy (Solove 2006)



Technical solutions' initial (main) focus: identification (≈ Orwell's methaphor)

Emerging issue: *correlations* (≈ Kafka's methaphor)

Legal (partial) solutions

- "Fair information practices" such as
  - Existence of personal data storage cannot be secret
  - Individuals can access their data (know usage)
  - Data minimization principle
  - Need of consent(opt in / out data collection)
  - Individual information can be corrected/Removed
  - Automated (pseudo) decisions can be contested
  - Organizations are responsible of data leaks
  - ...
- Not always easy to implement/quantify/verify

Rest of the lecture...

- Three different privacy enhancing models
  - Model 1: open (sanitized) data
    - Pseudonyms and limitations
    - k-anonymity, extensions and limitations
    - IT metrics and limitations
    - Extension: from Orwell to Kafka (& utility)
  - Model 2: trusted DB with sanitized answers
  - Model 3: Untrusted DB (MPC, FHE and limitations)
- Ranging from most convenient to use but least private to most private but least convenient to use
- All with (strong) limitations (no perfect solutions)

Vocabulary

- Identifier: piece of info. that identifies a person
  - e.g., name, address, passport number, ...
- Attributes: piece of info. that correlates with a person
  - Can be deterministic or probabilistic
    - e.g., gender (D) vs. movie preferences (P)
  - Can be sensitive or non-sensitive
    - e.g., medical prescriptions (S) vs. age (A)
- Leakage (sometimes auxiliary information): piece of information (usually non-sensitive) that can be used for re-identification or access to sensitive data

Pseudonyms

- Example of "pseudonymized" deterministic database

pseudo.	job	sex	age	disease
U1	engineer	m	35	hepatitis
U2	engineer	m	38	hepatitis
U3	lawyer	m	38	HIV
U4	writer	f	30	flu
U5	writer	f	30	HIV
U6	dancer	f	30	HIV
U7	dancer	f	30	HIV

sensitive data

- How strong are pseudonyms if we learn/leak that:
  - Bob is a 38 yo, male engineer in the DB?
    - Reveal all info

Pseudonyms

- Example of "pseudonymized" deterministic database

pseudo.	job	sex	age	disease
U1	engineer	m	35	hepatitis
U2	engineer	m	38	hepatitis
U3	lawyer	m	38	HIV
U4	writer	f	30	flu
U5	writer	f	30	HIV
U6	dancer	f	30	HIV
U7	dancer	f	30	HIV

- How strong are pseudonyms if we learn/leak that:
  - Cathy is a 30 yo, female writer in the DB?
    - u4 or u5

Pseudonyms

- Example of "pseudonymized" deterministic database

pseudo.	job	sex	age	disease
U1	engineer	m	35	hepatitis
U2	engineer	m	38	hepatitis
U3	lawyer	m	38	HIV
U4	writer	f	30	flu
U5	writer	f	30	HIV
U6	dancer	f	30	HIV
U7	dancer	f	30	HIV

- How strong are pseudonyms if we learn/leak that:
  - Emily is a 30 yo, female dancer in the DB?
    - HIV and U6 or U7

Pseudonyms

- Example of "pseudonymized" deterministic database

pseudo.	job	sex	age	disease
u1	engineer	m	35	hepatitis
u2	engineer	m	38	hepatitis
u3	lawyer	m	38	HIV
u4	writer	f	30	flu
u5	writer	f	30	HIV
u6	dancer	f	30	HIV
u7	dancer	f	30	HIV

- Would you prefer being Cathy or Emily? {...}
- How would you improve the situation? {...}

k-anonymity (I)

- Aims at preventing linkage on pseudonyms
- e.g., thanks to information leakage (or simply non-anonymized/non-sensitive databases)
  - Let *o* be a row of the database, *u* be a user
  - Define *anonymity set*: $\mathcal{A}(o) = \{u | \mathrm{DB}(u,:) = o\}$
  - *k*-anonymity is provided for

$$k = \min_{o \in \mathrm{DB}} |\mathcal{A}(o)|$$

Set of user such that they have the exatly same non sensitive attribute

- Typically achieved with 3 technical tools, namely
  - Aggregation of attributes
  - Grouping users
  - Noise addition (on data)

k-anonymity (II)

- Example of "*k*-anonymized" deterministic database

pseudo.	job	sex	age	disease
u1	professional	m	35-40	hepatitis
u2	professional	m	35-40	hepatitis
u3	professional	m	35-40	HIV
u4	artist	f	30-35	flu
u5	artist	f	30-35	HIV
u6	artist	f	30-35	HIV
u7	artist	f	30-35	HIV

k=3

k=4

- Which tool does it use? Aggregation of attributes
- What is the value of *k*? regarder ligne, minimum combien ont les mêmes infos ? => 3
- What is the main limitation? {...}

l-diversity (I)

- *k*-anonymity does not provide protection against a lack of diversity in an anonymity set
- *l*-diversity requires that that the sensitive attributes are "well distributed" in each anonymity set
- Basic version: at least *l* distinct values in each $\mathcal{A}(o)$
- Many variants exist. What would be the ideal?
  -

l-diversity (II)

pseudo.	job	sex	age	disease
u1	professional	m	35-40	hepatitis
u2	professional	m	35-40	hepatitis
u3	professional	m	35-40	HIV
u4	artist	f	30-35	flu
u5	artist	f	30-35	HIV
u6	artist	f	30-35	HIV
u7	artist	f	30-35	HIV

- What is the value of l? 2
- How many records needed to reach $l = 3$? → add 2 rows
- How many records needed to reach ideal setting? 8
- What are the main limitations of this setting? {...}

All is discrete

Interlude: t-closeness

- Formalizes a relaxation of the "ideal setting"
- The distribution of sensitive attributes in an anonymity set is "close" to the distribution of the full table
  - Close is captured by a statistical distance
- We will discuss it together with the second model

Probabilistic databases

pseudo.	comedy	drama
u_1	... **65** (*truth 60%*)	... **35** (*truth 40%*)
u_2	... **50** (*truth 55%*)	... **50** (*truth 45%*)

truth distribution

measurement

- Say $o^1_{\text{Bob}} = \text{comedy}$, what is $\tilde{\Pr}[U = u_1 | O = o^1_{\text{Bob}}]$?
  - With o^1_{Bob} an internal leakage from the DB
- And $\tilde{\Pr}[U = u_1 | o^1_{\text{Bob}}, o^2_{\text{Bob}}]$ with $o^2_{\text{Bob}} = \text{comedy}$?
- And $\tilde{\Pr}[U = u_1 | o^1_{\text{Bob}}, o^2_{\text{Bob}}, \ldots, o^{100}_{\text{Bob}}]$? (i.e., full DB)
- "Continuous attacks" not captured by k-anonymity / l-diversity. For example, $\forall o \in \text{DB}, |\mathcal{A}(o)| =$

Probabilistic databases

pseudo.	comedy	drama
u_1	... **65** (*truth 60%*)	... **35** (*truth 40%*)
u_2	... **50** (*truth 55%*)	... **50** (*truth 45%*)

- Say $o^1_{\text{Bob}} = \text{comedy}$, what is $\tilde{\Pr}[U = u_1 | O = o^1_{\text{Bob}}]$?
  - With o^1_{Bob} an internal leakage from the DB
- And $\tilde{\Pr}[U = u_1 | o^1_{\text{Bob}}, o^2_{\text{Bob}}]$ with $o^2_{\text{Bob}} = \text{comedy}$?
- And $\tilde{\Pr}[U = u_1 | o^1_{\text{Bob}}, o^2_{\text{Bob}}, \ldots, o^{100}_{\text{Bob}}]$? (i.e., full DB)
- Reminder: k-anonymity and l-diversity capture "single-shot attacks" in an "all-or-nothing" fashion

Probabilistic databases

pseudo.	comedy	drama
u_1	... **65** (*truth 60%*)	... **35** (*truth 40%*)
u_2	... **50** (*truth 55%*)	... **50** (*truth 45%*)

- Say $o^1_{\text{Bob}} = \text{comedy}$, what is $\tilde{\Pr}[U = u_1 | O = o^1_{\text{Bob}}]$?
  - With o^1_{Bob} an internal leakage from the DB
- And $\tilde{\Pr}[U = u_1 | o^1_{\text{Bob}}, o^2_{\text{Bob}}]$ with $o^2_{\text{Bob}} = \text{comedy}$?
- And $\tilde{\Pr}[U = u_1 | o^1_{\text{Bob}}, o^2_{\text{Bob}}, \ldots, o^{100}_{\text{Bob}}]$? (i.e., full DB)
- Continuous attacks quite similar to "side-channel attacks" against crypto. devices (see ELEC2760)

= 1 +u1. 1/2 (0,65.log(0,65/0,65+0,5) + 0,35 . log(0,55/0,35+0,5)) + u2 . 1/2 (comedy + drama)

(Hypothetical) Information

All User All DB

$$\tilde{\text{HI}}(U; O) = \text{H}[U] + \sum_{u \in \mathcal{U}} \Pr[u] \cdot \sum_{o \in \text{DB}} \tilde{\Pr}[o|u] \cdot \log_2 \tilde{\Pr}[u|o],$$

- with $\Pr[X = x] := \Pr[x]$ & $\text{H}[U] = -\sum_u \Pr[u] \cdot \log_2 \Pr[u]$

Entropy of the user (how random the data is)

- Example with the previous database

log(# of user)

pseudo.	comedy	drama
u_1	... **65** (*truth 60%*)	... **35** (*truth 40%*)
u_2	... **50** (*truth 55%*)	... **50** (*truth 45%*)

Average bias

- How much is worth $\tilde{\text{HI}}(U; O)$ (& why hypothetical)?

Main theorem (informal)

- Given $\tilde{\mathrm{HI}}(u; O)$ (for one user), and *n internal* leakages from the DB, the re-identification success rate

powerful

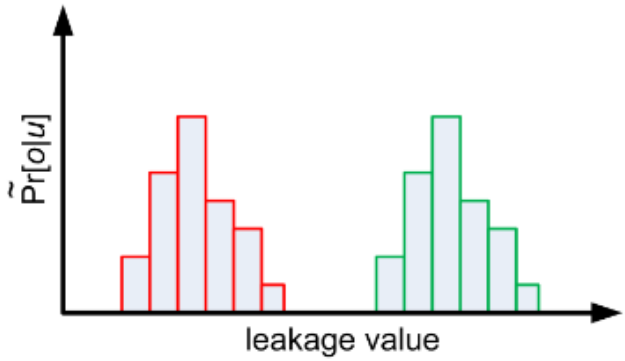
$$\mathbf{SR}_{\mathrm{id}}(n) \leq 1 - \left(1 - \tilde{\mathrm{HI}}(u; O)\right)^n$$

- Under several assumptions (!), most critically
  - Leakages are picked up uniformly from the DB
  - Distributions are "kind" (i.e., stable and noisy)
- Number of leakages to reach high success rate

$$N = \frac{L}{H_1} \geq \left[\frac{1}{\tilde{\mathrm{HI}}(u, O)}\right]$$

Intuition

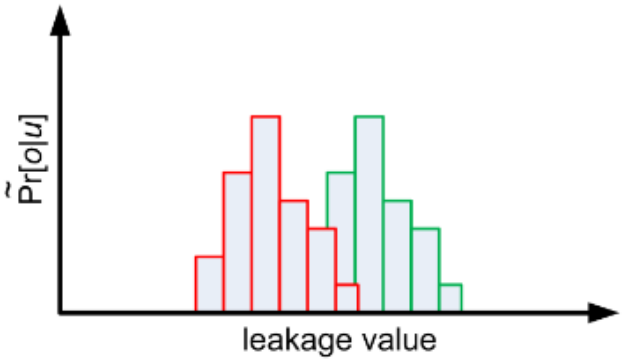
- Consider a 2-user case study with $\tilde{\Pr}[o|u]$



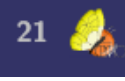
- It leads to a Hypothetical Information $\tilde{\mathrm{HI}}(U; O) = 1$

Intuition

- Consider a 2-user case study with $\tilde{\Pr}[o|u]$

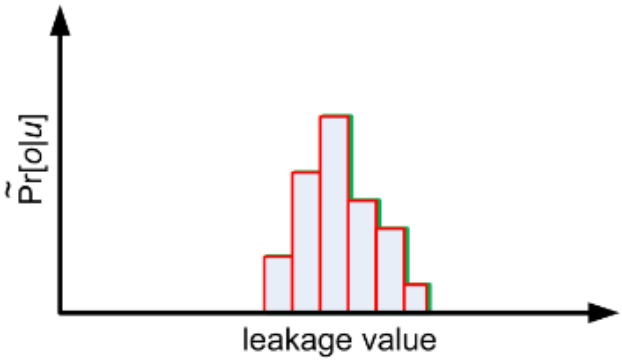


- It leads to a Hypothetical Information $\tilde{\mathrm{HI}}(U;O) =$



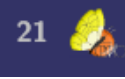
Intuition

- Consider a 2-user case study with $\tilde{\Pr}[o|u]$



We want

- It leads to a Hypothetical Information $\tilde{\mathrm{HI}}(U;O) = 0$



Cautionary notes

- Analysis per user since they can have very different distributions (more or less discriminating)
- Assumptions most likely do not hold for actual databases (actual user's data does not come from a regular distribution and is not enough noisy)

- Yet, general intuition is correct: users' privacy decreases exponentially with the number of leakages
  - Privacy vanishes fast in open data publishing
  - & things get worse if multiple DBs can be combined

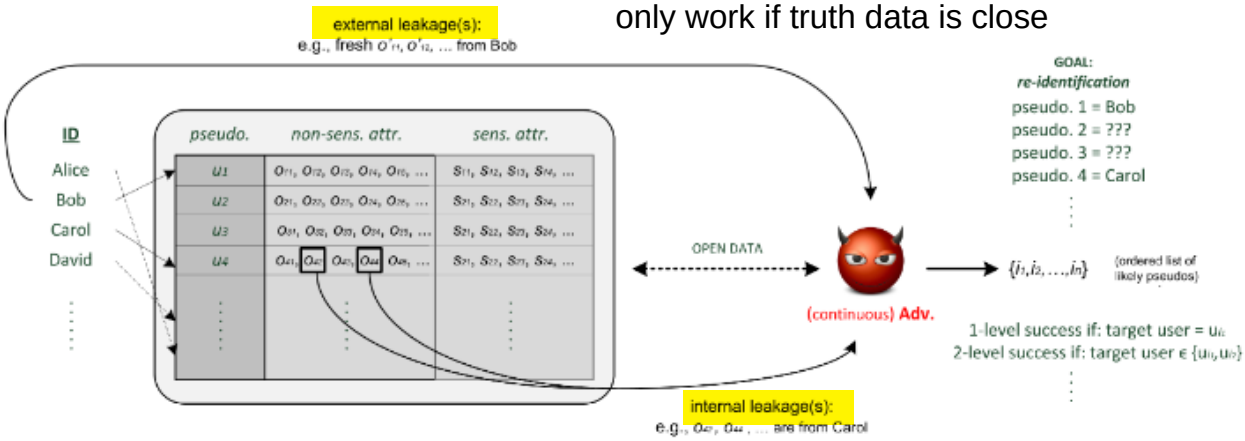
From Orwell to Kafka

Kafka's metaphor: surreal nightmare of a person unexpectedly informed that he is under arrest but given no reason why

- Looks far sketched? Think about an insurance company giving you a higher-fare policy because of some of your preferences it found on the Internet
- Or a job refused because of social networking
- Legal viewpoint: users should be aware of stat. proc.

Technical viewpoint

- Leakages can also be *external* to the DB



⇒ Re-ident. then depends on whether the DB is predictive

Perceived Information

- Split the DB in two parts: DB_p and DB_t
  - Where p stands for profiling and t for testing
- Build a model $\tilde{Pr}[o|u]$ ← DB_p for every user

$$PI(U;O) = H[U] + \sum_{u \in \mathcal{U}} Pr[u] \cdot \sum_{o \in \mathcal{O}} \underbrace{Pr[o|u]}_{true} \cdot \log_2 \underbrace{\tilde{Pr}[u|o]}_{model} \cdot$$

- ≈ statistical distance between the model and true distri.

Perceived Information

- Split the DB in two parts: DB_p and DB_t
  - Where p stands for profiling and t for testing
- Build a model $\tilde{Pr}[o|u] \leftarrow DB_p$ for every user

$$\hat{PI}(U;O) = H[U] + \sum_{u \in \mathcal{U}} Pr[u] \cdot \sum_{o \in DB_t} \frac{1}{|DB_t|} \cdot \log_2 \underbrace{\tilde{Pr}[u|o]}_{model} \cdot$$

- But true distri. unknown $\Rightarrow$ can only be sampled
- Intuitively: amount of information that can be extracted from obs. biased by model errors

Examples

pseudo.	DB_p		DB_t	
	comedy	drama	comedy	drama
u1	... **299** (*truth 60%*)	... **201** (*truth 40%*)	... **600** (*truth 60%*)	... **400** (*truth 40%*)
u2	... **251** (*truth 55%*)	... **249** (*truth 45%*)	... **500** (*truth 55%*)	... **500** (*truth 45%*)

on augmente le nb d'échantillon => meiilleur distribution

$H(U) = 1 = \log(2)$

- With $|DB_p| = 100$, $\hat{PI}(U;O) =$ 0, impossible de les distinguer
- What is the value of $\hat{PI}(U;O)$ with $|DB_p| = 500$

$$\hat{PI}(U,O) = 1 + \frac{1}{2}\left(0{,}6 \cdot \log\left(\frac{110}{110+105}\right) + 0{,}4 \cdot \log\left(\frac{50}{50+95}\right)\right) + \frac{1}{2}\left(0{,}55 \log\left(\frac{105}{110+105}\right) + 0{,}45 \cdot \log\left(\frac{95}{50+95}\right)\right)$$

Examples

pseudo.	DB_p		DB_t	
	comedy	drama	comedy	drama
u1	... **299** (*truth 60%*)	... **201** (*truth 40%*)	... **600** (*truth 60%*)	... **400** (*truth 40%*)
u2	... **251** (*truth 55%*)	... **249** (*truth 45%*)	... **500** (*truth 55%*)	... **500** (*truth 45%*)

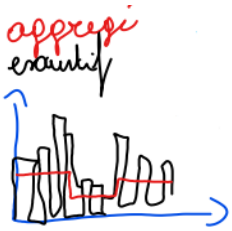
- With $|DB_p| = 100$, $\hat{PI}(U; O) =$
- What is the value of $\hat{PI}(U; O)$ with $|DB_p| = 500$
- As the size of DB_p increases, the model better reflects /predicts the true distribution $\Rightarrow$ re-identification attacks with external leakages become more effective

Summary (I)

- Re-identification with external leakages also possible
- But the biggest concern in this case is inference
- Positive PI means the collected data is "useful"
  - i.e., is correlated with the true user's distributions

on nous identifies a un ensemble (eg assurance

$\Rightarrow$ "pseudo-decisions" can be made by autonomous systems



- Worst-case PI hard (if not impossible) to evaluate
  - Optimal models can be (very/too) slow to converge
  - Most models trade estimation & assumption errors
    - 2-user example with Matlab code (test_PI.m)

Summary (II)

- Anonymization techniques do not help!
  - e.g., correlation with a group model is enough
- Suggests technical limits for open data publishing
- Solution 1 (next): trusted DB with sanitized answers (but strong trust assumption: the operator is not using the data for other means *and is able to protect it!*)
- Solution 2 (last): MPC or FHE (but puts strong constraints on the private statistical treatment)

faisable

Cautionary note

- Utility is in general hard to define too
- Most standard options are twofold
  - General purpose metrics
    - e.g., minimal distortion of the sanitized DB
    - But assumes any data detail is useful
  - Specialized metrics
    - Based on the purpose of the data collection
    - e.g., sanitized DB "preserve" the attributes' mean
- PI definition of utility is still under investigation

data reflects PDF +/- usefull data

Towards Model 2

- Model 1 is too liberal w.r.t. privacy
- How can we reduce the Adv.'s power?
  - only access to output of queries
- Ideal setting re-stated:

access to outputs should not enable to learn anuthing about an individual (that could not be learned without access to the db

Not really possible => add noise

Example

pseudo.	smoker	cancer
u_1	Y	Y
u_2	Y	N
u_3	N	N
u_4	N	N
u_5	Y	Y
u_6	N	N
u_7	N	Y
...	...	...

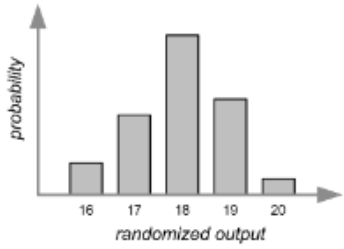
Cancer is correlated with smoking ($\rho = 0.18$)

- Learning such a correlation is legitimate
- But what if we learn Alice smokes? $\{\dots\}$

$\Rightarrow$ We need a way to trade utility & privacy
(*anything is too strong in the previous ideal setting*)

Tool: noise addition

pseudo.	smoker	cancer
u1	Y	Y
u2	Y	N
u3	N	N
u4	N	N
u5	Y	Y
u6	N	N
u7	N	Y
...	...	...



not continous data

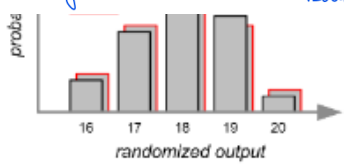
- More realistic promise: if you leave the database, the (randomized) outcome will not change much
- So why leaving the experiment in this case?

Differential Privacy

pseudo.	smoker	cancer
u1	Y	Y
u2	Y	N
u3	N	N
u4	N	N
u5	Y	Y
u6	N	N
u7	N	Y
...	...	...

- ϵ-differential privacy for mechanism M if for any two neighboring DBs D_1, D_2 and any $c \in \text{range}(M)$

$$\Pr[M(D_1) = c] \leq e^{\epsilon} \Pr[M(D_2) = c]$$



- (ϵ, δ)-differential privacy for mechanism M if for any two neighboring DBs D_1, D_2 and any $c \in \text{range}(M)$

$$\Pr[M(D_1) = c] \leq (1 + \epsilon) \Pr[M(D_2) = c] + \delta$$

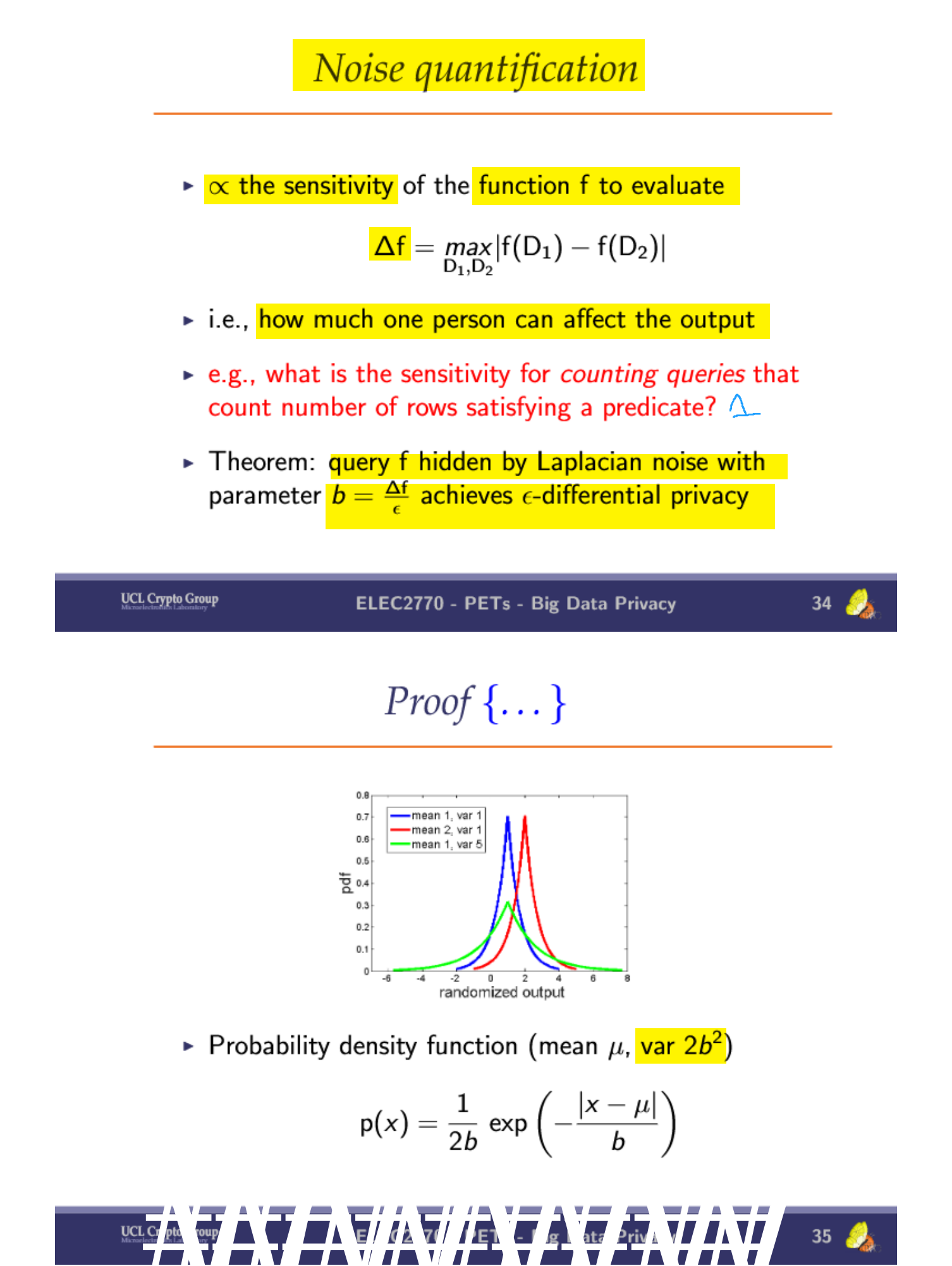
(*Intuitively: to deal with the outputs with* $\Pr = 0$)

Noise quantification

- $\propto$ the sensitivity of the function f to evaluate

$$\Delta f = \max_{D_1, D_2} |f(D_1) - f(D_2)|$$

- i.e., how much one person can affect the output
- e.g., what is the sensitivity for *counting queries* that count number of rows satisfying a predicate? 1
- Theorem: query f hidden by Laplacian noise with parameter $b = \frac{\Delta f}{\epsilon}$ achieves ϵ-differential privacy



Proof {...}

- Probability density function (mean μ, var $2b^2$)

$$p(x) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$

Advantages

- DP is a property of the mechanism (not the data)
  - Unaffected by leakages / auxiliary information
- Composition theorem: an ϵ_1-DP mechanism followed by an ϵ_2-DP mechanism is $(\epsilon_1 + \epsilon_2)$-DP
  - Similar to exp. security decrease with IT metrics
  - But in a worst-case (rather than avg.) setting!
- Link with t-closeness: $\exp(\epsilon/2)$-cloneness $\Rightarrow \epsilon$-DP

Non-trivial improvements

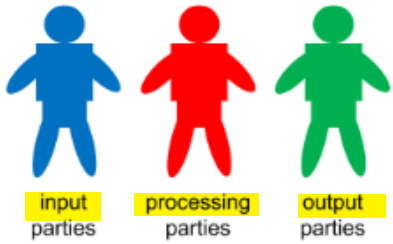
- For example, think about "histogram queries"
  - i.e., we want to know the # of people with ages [0-4], [5-9], [10-14], ... with a total of q bins
- What is the sensitivity of 1 bin?
- How much noise needed for q queries? Lap(...)
- In fact, Lap($\frac{1}{\epsilon}$) is enough! (Intuition: if one person leaves the database, only one bin can change)
- Many other results in the literature

Summary

- Main advantage/interest of DP is to limit the adversarial power thanks to an interactive setting
- DP theoretically allows strong formal guarantees BUT
  - It is based on a strong trust assumption
  - The noise levels to reach meaningful security levels are frequently prohibitive w.r.t. data utility
- Next: what can be done if without trusting the DB

(*Start with the preliminary reading by Bringer et al.*)

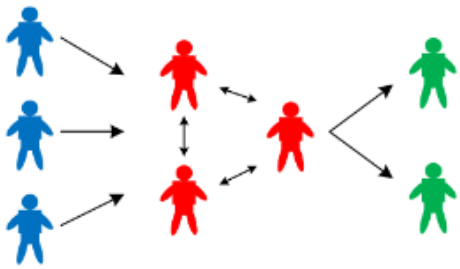
Untrusted DB



Traditional setting: one of each and they are all the same person

- What if we separate input, proc. and output parties?
  - e.g., in differential privacy, the processing is performed (in clear) by a trusted 3rd party
- MPC/FHE: how to compute on private data?

Multi-Party Computation



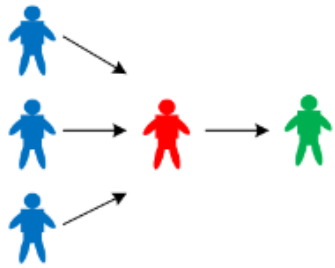
- Several input, processing and output parties
- Processing parties can be semi-honest, malicious, ...
- Previous lectures: MPC based on garbled circuits
- Next lecture: MPC based on secret sharing

Bringer et al.

- What is the goal of biometric identification? {...}
- How many input/output parties does it involve? 2
- How to obtain the eigenface repr. of an image? {...}
- Why can profiling be excluded from the MPC? {...}
- Is the projection (eigenvectors) sensitive? {...} Yes
- Why is eigenf. repr. a good candidate for MPC? {...}
- How to protect against a dishonest out. party? JP

Euclidian distance

Fully Homomorphic Encryption

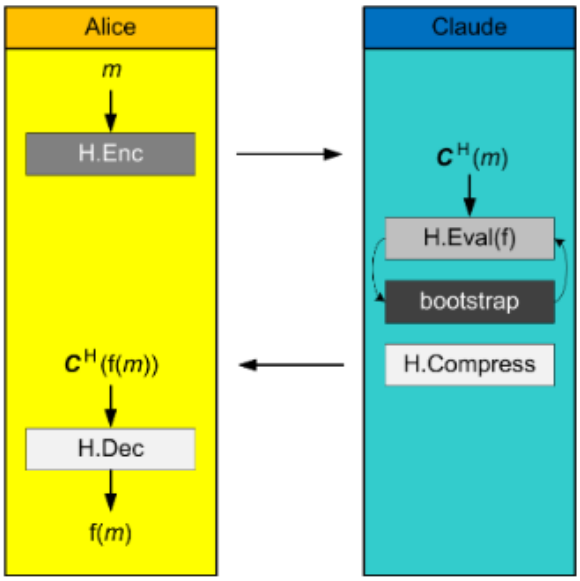


- One processing party (!), one or many input parties, usually one output party (but could be more)
- To protect against malicious computing party, need to add a *verifiable computation protocol* on top
- (+ DP needed if the output party is dishonest)

more gray = more computationaly intensive

S = symmetric
H = homomorphic

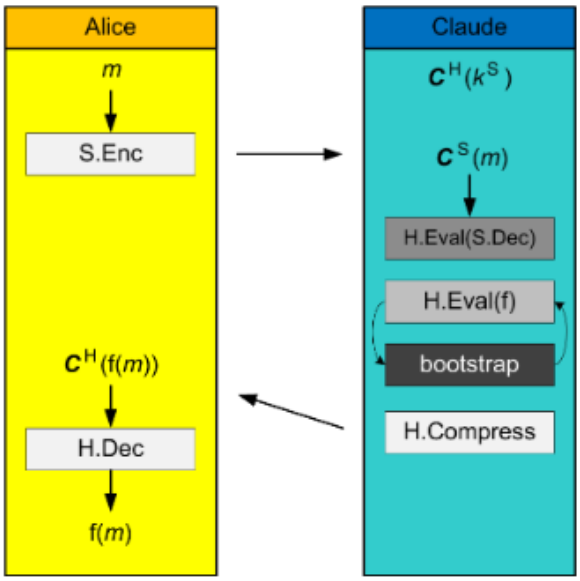
FHE framework



Msg. m never appears in clear to Claude

Gray scale ≈ cost

FHE-SE hybrid framework



Cheaper for Alice

(+ bootstrap skipped if S.Enc & f simple)

FHE intuitions (I)

- Learning With Errors (LWE) assumption

$$\{\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{k} \rangle + e_i\}_{i=1}^{\text{poly}(n)} \overset{c}{\approx} \{\mathbf{a}_i, u_i\}_{i=1}^{\text{poly}(n)}$$

- For $\mathbf{a}_i$ and $\mathbf{k} \in \mathbb{Z}_q^n$ and e_i a noise sample
- LWE-based Enc: $c = \langle \mathbf{a}_i, \mathbf{k} \rangle + 2e + m$ in $\mathbb{Z}_q^n \times \mathbb{Z}_q$
- LWE-based Dec: $m = (\langle \mathbf{a}_i, \mathbf{k} \rangle + c \bmod q) \bmod 2$
  - Possible if $e \ll q$ (i.e., low enough noise)
- Define $f_{\mathbf{a},c}(\mathbf{x}) : \mathbb{Z}_q^n \to \mathbb{Z}_q := c - \langle \mathbf{a}_i, \mathbf{x} \rangle \bmod q$

FHE intuitions (II)

- Add. homomorphism: $f_{\mathbf{a}+\mathbf{a}',c+c'}(\mathbf{x}) = f_{\mathbf{a},c}(\mathbf{x}) + f_{\mathbf{a}',c'}(\mathbf{x})$
- If $\sigma^2(c) := \sigma^2(e) = \epsilon$, how much is $\sigma^2(c + c')$?
- Multiplicative homomorphism is a thorny problem

$$f_{\mathbf{a},c}(\mathbf{x}) \cdot f_{\mathbf{a}',c'}(\mathbf{x}) = \left(c - \sum \mathbf{a}[i]\,\mathbf{x}[i]\right) \cdot \left(c' - \sum \mathbf{a}'[i]\,\mathbf{x}'[i]\right)$$

- Problem 1: (basic homomorphic) decryption has to know the coefficients of a quadratic polynomial in the variables of $\mathbf{x}$ $\Rightarrow$ quadratic blowup of ciphertext size

FHE intuitions (II)

- Add. homomorphism: $f_{\mathbf{a}+\mathbf{a}',c+c'}(\mathbf{x}) = f_{\mathbf{a},c}(\mathbf{x}) + f_{\mathbf{a}',c'}(\mathbf{x})$
- If $\sigma^2(c) := \sigma^2(e) = \epsilon$, how much is $\sigma^2(c + c')$?
- Multiplicative homomorphism is a thorny problem

$$f_{\mathbf{a},c}(\mathbf{x}) \cdot f_{\mathbf{a}',c'}(\mathbf{x}) = \left(c - \sum \mathbf{a}[i]\,\mathbf{x}[i]\right) \cdot \left(c' - \sum \mathbf{a}'[i]\,\mathbf{x}'[i]\right)$$

- Problem 2: If $\sigma^2(c) = \epsilon$, how much is $\sigma^2(c \cdot c')$?
  $\Rightarrow$ ciphertexts are rapidly too noisy to be decrypted
  (*Re-linearization/bootstrapping prevent these issues*)

rapid to noisy => bootstrap => refresh ciphertext => reduce noise

No interaction

FHE intuitions (II)

- Add. homomorphism: $f_{\mathbf{a}+\mathbf{a}',c+c'}(\mathbf{x}) = f_{\mathbf{a},c}(\mathbf{x}) + f_{\mathbf{a}',c'}(\mathbf{x})$
- If $\sigma^2(c) := \sigma^2(e) = \epsilon$, how much is $\sigma^2(c + c')$?
- Multiplicative homomorphism is a thorny problem

$$f_{\mathbf{a},c}(\mathbf{x}) \cdot f_{\mathbf{a}',c'}(\mathbf{x}) = \left(c - \sum \mathbf{a}[i]\,\mathbf{x}[i]\right) \cdot \left(c' - \sum \mathbf{a}'[i]\,\mathbf{x}'[i]\right)$$

- Problem 2: If $\sigma^2(c) = \epsilon$, how much is $\sigma^2(c \cdot c')$?
  $\Rightarrow$ ciphertexts are rapidly too noisy to be decrypted
  (*But are highly techical and so far extremely expensive*)
- How does MPC deal with this issue?
  ! add interactions, compute together !

Summary

- In practice, (current) MPC is much more efficient than FHE (e.g., FHE key sizes in the Gbit range)
- Both tools work in very constraint settings (far away from the "collect & see" paradigm in Big Data)
- Typical annoying question (in MPC/FHE): who can decrypt? $\Rightarrow$ Makes them quite application-dependent
- Main challenge: how to protect model estimation? (Most MPC/FHE solution focus on model evaluation)

Conclusions

- Data privacy is an important practical problem
  - But a quite hard to define one!
- There are technical solutions to improve various definitions of privacy; but none is fully satisfying
- Unfortunately, better guarantees generally come at the cost of either a reduced data utility or limiting the adversary (hence reducing flexibility/versatility)

Privacy Enhancing Technologies

O. Pereira and F.-X. Standaert

UCL Crypto Group, Université catholique de Louvain

Mass Surveillance, Post-Snowden Crypto and Hardware Trojans (2 lectures)

Outline

- Introduction / motivation
- Cryptovirology / Kleptography
  - The case of NIST's "Dual EC" PRNG
- Algorithm Substitution Attacks
- Cryptographic Reverse firewalls
- Big key (symmetric) cryptography
- Hardware Trojans (device corruption)
  - Trojan-resilient circuits from MPC

Snowden revelations (2013)

NSA (and others): "know it all, collect it all, exploit it all"

- Server side: target plaintext or key
  - Slight surprise: industry (has to) collaborate(s)
- Communication: target data & meta-data
- Client side: active hacking (even in supply chain)

Impact

- Theoretical viewpoint: Dolev-Yao model is real
  - Omnipotent adversary who "carries the messages"
  - While considered quite impractical when introduced
- Practical viewpoint (societal impact)
  - From phone number, get: personal devices, web surfing behavior, locations, collaborators, ...
  - But only collecting all meta-data is considered disproportionate by the EU Court of Justice (2014)!
- Note: Google probably knows even more about you

Another crypto failure?

crypto. adversaries:
limited resources / obey rules

real adversaries:
huge resources

Another crypto failure?

crypto. adversaries:
limited resources / obey rules

real adversaries:
huge resources / don't obey rules

- In some sense nothing new (see ELEC2760)
- But the scale is surprisingly broad

Rest of the lecture...

- Investigation of technical issues & partial solutions
  - Exploiting algorithms & implementations
  - Ranging from specific to generic
- Undermined standards (Kleptography)
- Undermined implementations targeting
  - Symmetric crypto (alg. substitution attacks)
  - Asymmetric crypto (reverse firewalls)
- Key exfiltration (big key crypto)
- Hardware Trojans (MPC-based solution)

Cryptovirology

Use of cryptography to design malicious software (viruses)

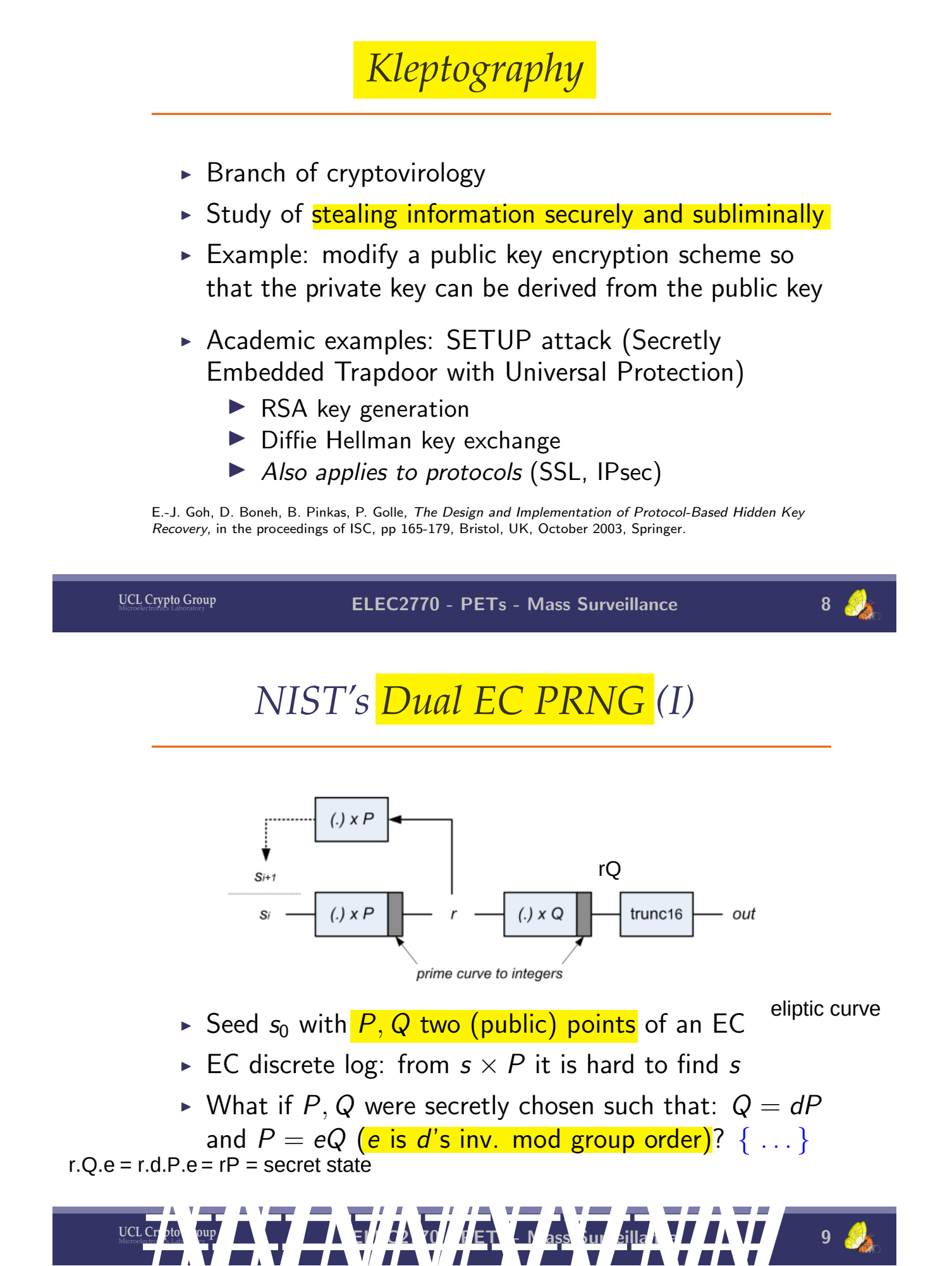
- Example: ransomware (aka cryptoworm, cryptovirus)
  - The malware encrypts the user's hard disk using the adversary's public key and asks a fee for decryption
  - It actually happened (e.g., in 2016, to a hospital)
  - Would it work with symmetric encryption? {...}

not rly, search the key and decrypt

Kleptography

- Branch of cryptovirology
- Study of stealing information securely and subliminally
- Example: modify a public key encryption scheme so that the private key can be derived from the public key
- Academic examples: SETUP attack (Secretly Embedded Trapdoor with Universal Protection)
  - RSA key generation
  - Diffie Hellman key exchange
  - *Also applies to protocols* (SSL, IPsec)

E.-J. Goh, D. Boneh, B. Pinkas, P. Golle, *The Design and Implementation of Protocol-Based Hidden Key Recovery*, in the proceedings of ISC, pp 165-179, Bristol, UK, October 2003, Springer.



NIST's Dual EC PRNG (I)

- Seed s_0 with P, Q two (public) points of an EC
- EC discrete log: from $s \times P$ it is hard to find s
- What if P, Q were secretly chosen such that: $Q = dP$ and $P = eQ$ (e is d's inv. mod group order)? { . . . }

eliptic curve

r.Q.e = r.d.P.e = rP = secret state

NIST's Dual EC PRNG (II)

- Snowden leaks suggest NSA backdoored the PRNG
- At least it is possible (and hard to detect)
- Solutions?
  - Truncate more than 16
  - Explain how P and Q are chosen
  - Don't use ECC for PRNG (expensive, slow)

Summary

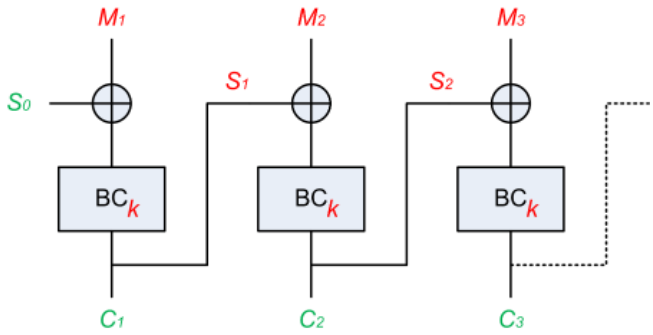
- Public key crypto brings great functionalities but also creates risks & opportunities for adversaries
- Dual EC PRNG highlights the need of transparent and well understood standards (and parameters)
- For example, see the 1M$ curve proposal
- Or think about the difference between DES and AES

Algorithm Substitution Attacks

- Assume our algorithms cannot be trapdoored
- The next natural step is to backdoor implementations
  - e.g., standard Symmetric Encryption (SE) schemes
- Algorithmic Substitution Attacks (ASAs)
  - Big brother adversary $\mathcal{B}$ substitutes the encryption scheme E by subverted one $\tilde{E}$
    - Typically means the software of E is corrupted
  - Ciphertexts from E and $\tilde{E}$ look similar to reg. users
  - Ciphertexts generated by $\tilde{E}$ leak (e.g., the key) to $\mathcal{B}$
- Can be viewed as a kind of kleptographic attack

BC = block cipher
k = mastery key
2 msg != ciphertexts

Attack against CBC



- $E := (S_0 \xleftarrow{r} \{0,1\}^n, C_i := BC_k(M_i \oplus S_{i-1}), S_i := C_i)$
- How to mount an ASA against this E?

$S_0 = BC_k^{-1}(k)$

looks the same but enemy can recover the k

Security against ASAs

- Semi-formal definition as follows
  - Detection adversary: models ordinary users who own K but not $\tilde{K}$ and want to know if ASA is underway
  - Surveillance adversary: models big brother who owns $\tilde{K}$ but not K and want to read users' traffic
  - Security if either detection adversary trivially succeeds or surveillance adversary miserably fails
- Why is CBC weak against ASAs? { . . . } detection of asas is hard
- Can be formalized as an indistinguishability game

Negative result

- Any randomized stateless SE scheme is vulnerable to an undetectable ASA leading to key recovery
- Proof idea based on the following attack
  - Let $\mathsf{F} : \{0,1\}^n \to \{0,1\}$ be a PRF with key $\tilde{K}$
  - To leak bit $K[j]$ of K, just encrypt with genuine seeds to get $C = \mathsf{E}_K(S_0, M)$ until $\mathsf{F}_{\tilde{K}}(C, j) = K[j]$ holds
- Main observation: a genuine seed is not enough

Defeating ASAs

- Randomized schemes are now bad...
- But stateless deterministic schemes (in general) cannot achieve semantic security (as usually required)
- Only option: use stateful deterministic schemes such that for any key K, message M and state t, of the decr algo, there is at most one

Instantiation (e.g., for Auth. Enc.)

$\underline{\mathsf{E}(K, M, A, \sigma)}$
If $(\sigma = 2^\ell)$ return $\perp$
$K_1||K_2 \leftarrow K$;
$W \leftarrow \mathsf{BC}_{K_1}(\sigma||M)$;
$T \leftarrow \mathsf{F}_{K_2}(W||A)$;
$C \leftarrow (W, T)$;
$\sigma \leftarrow \sigma + 1$;
Return (C, σ);

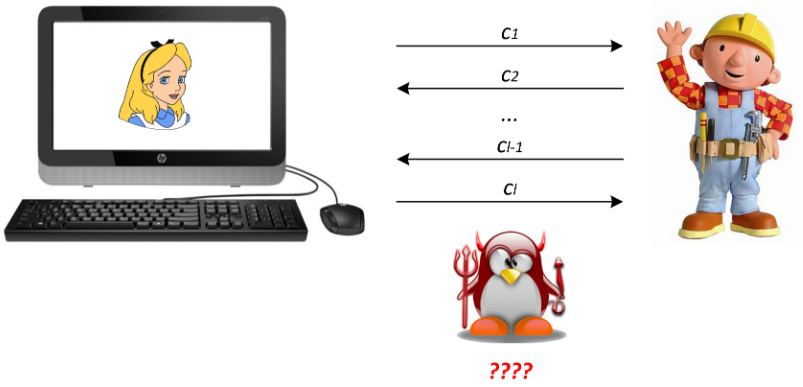
$\underline{\mathsf{D}(K, C, A, \tau)}$
If $(\tau = 2^\ell)$ return $\perp$
$K_1||K_2 \leftarrow K$; $(W, T) \leftarrow C$;
$(\sigma, M) \leftarrow \mathsf{BC}^{-1}_{K_1}(W)$;
If $(T \neq \mathsf{F}_{K_2}(W||A))$ return $\perp$
If $(\sigma \neq \tau)$ return $\perp$;
$\tau = \tau + 1$;
Return (M, τ);

- Doubly stateful scheme (σ and τ are counters)
- Reject permanently if decryption fails once

Summary

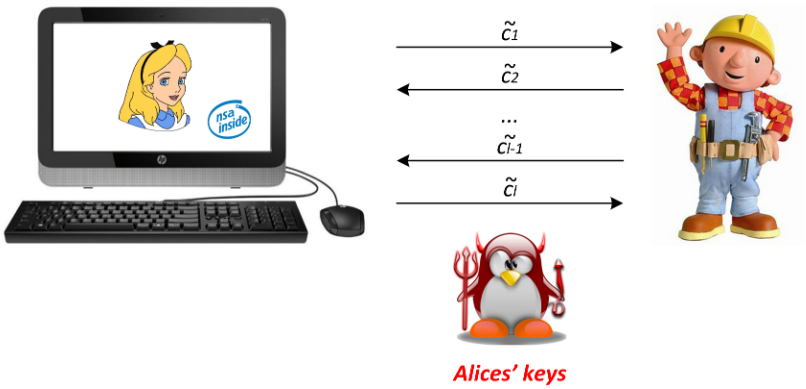
- ASAs are easy to implement with corrupted software
- RNGs are great targets for inserting backdoors
- Stateful deterministic schemes can help for SE
  - But not all of them (unique ciphertexts needed)
- Problem for Public Key Cryptography (see next)

Crypto. reverse firewalls



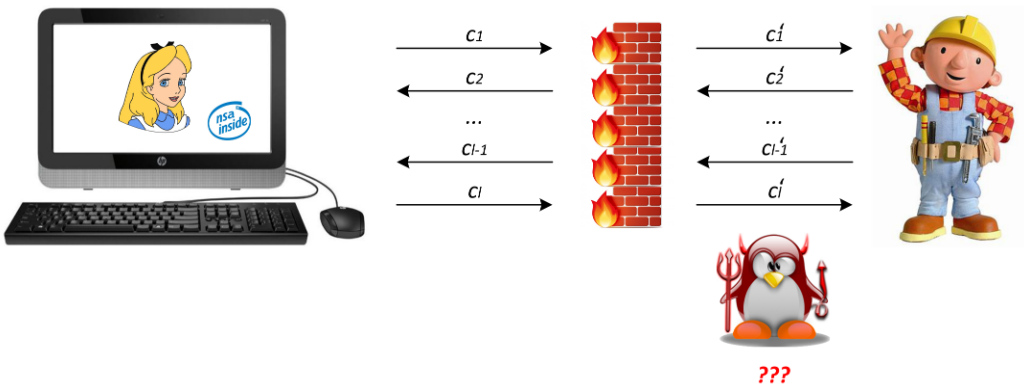
- Cryptography is sound if Alice's computer is trusted

Crypto. reverse firewalls



- But can fail dramatically otherwise (e.g., ASAs)

Crypto. reverse firewalls

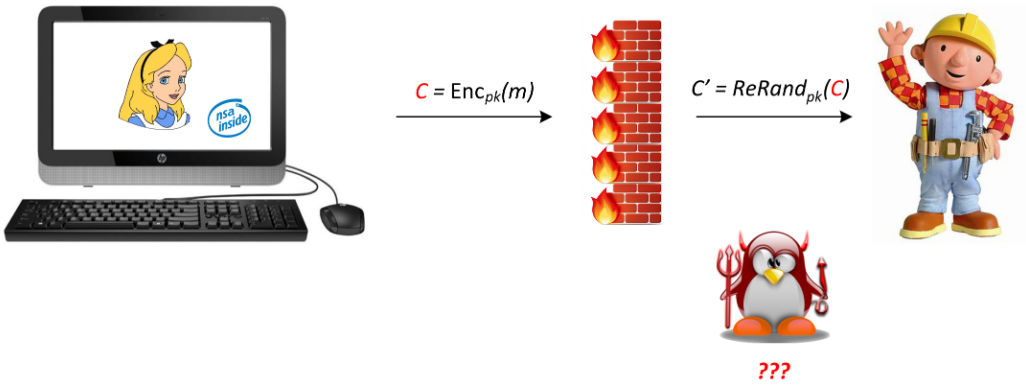


- Goal of the firewall: improve security without sharing additional secrets with Alice (especially for PKC)

Desirable features

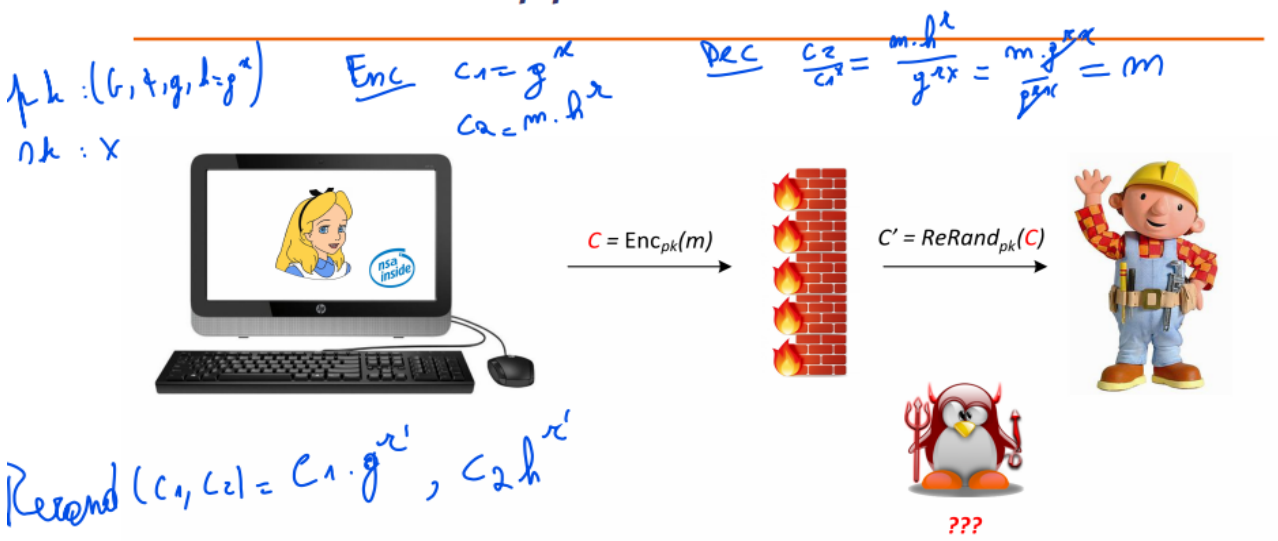
- For protocol P with functionality f satisfying an ideal security notion S, the firewall $\mathcal{W}$ should ensure
- Correct functionality: P + W has func f
- Security: P + W satisfies S
- Regardless the fact that Alice's computer is corrupted

Application #1



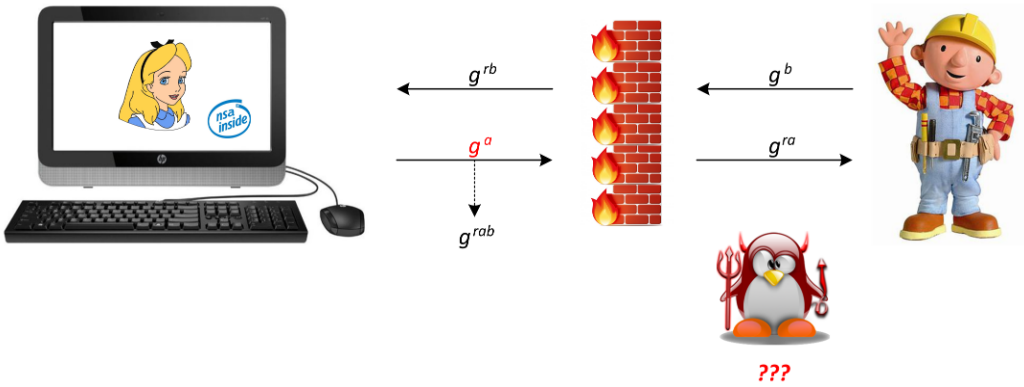
- Semantically secure if Enc_{pk} is a re-randomizable PKC
  - $\text{Dec}(\text{ReRand}(\text{Enc}(m))) = m$, $\text{ReRand}(C) \approx_c \text{Enc}(0)$

Application #1



- Semantically secure if Enc_{pk} is a re-randomizable PKC
  - Example of such a PKC? ElGamal

Application #2



- Diffie Hellman key exchange protocol
  - Is it secure (e.g., can Alice's computer bias the key)?
  - No (secure if Alice commit g^a)

Summary

- Cryptographic reverse firewalls counteract randomness-based trapdoors by re-randomizing
- Allows PKC (and more fancy primitives)
- But re-randomization is not trivial
- How to deal with a corrupted firewall? {...}

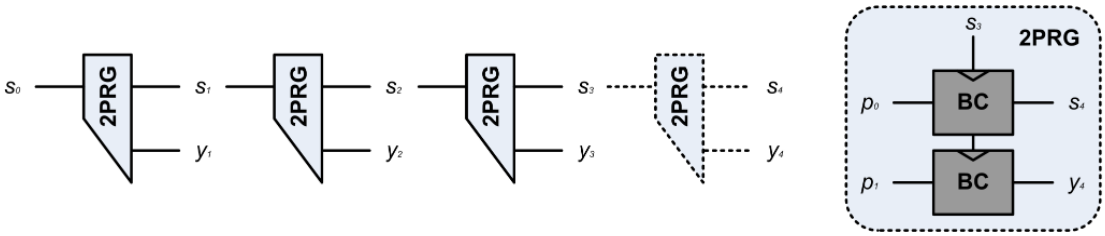
loose it but can add MORE walls

Key exfiltration

- Orthogonal concern: assume an adversary who can transmit the key via steganography (despite the previous algorithmic improvements)
- In other words, the key is leaked (but not fully)
- What can be done in this setting?

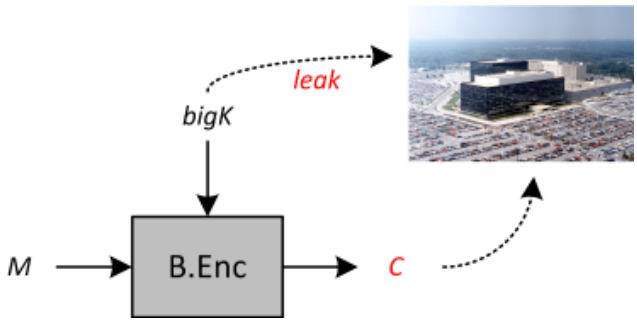
Adi Shamir (2013): *Advanced Persistent Threats have a very narrow pipeline to the outside world...*

Leakage-resilience



- Each iteration leaks l bits of the running seed
  - See ELEC2760 for the details
- Theorem: if the security of iteration is $\frac{1}{2^{k-l}}$ then security of q iterations is (ideally) $\approx$
- Problem: initial seed derived from constant master key

Big Key Cryptography (I)



- Typically: k-bit $bigK$ ($k \approx 10^9$), $|leak| \approx k/10$
- One-shot global (i.e., not continuous) leakage bound
  - $\neq$ side-channel attacks against embedded devices

Big Key Cryptography (II)

B.Enc (simplified)

$\{\text{probes}\} \xleftarrow{r} \{\text{key indices}\};$
$K = bigK(\text{probes});$
$C = \text{BC}_K(M);$
(*In fact requires RO*)

l = amount of bit leak
k = total bit amount

- Leads to (surprisingly hard to derive) bound

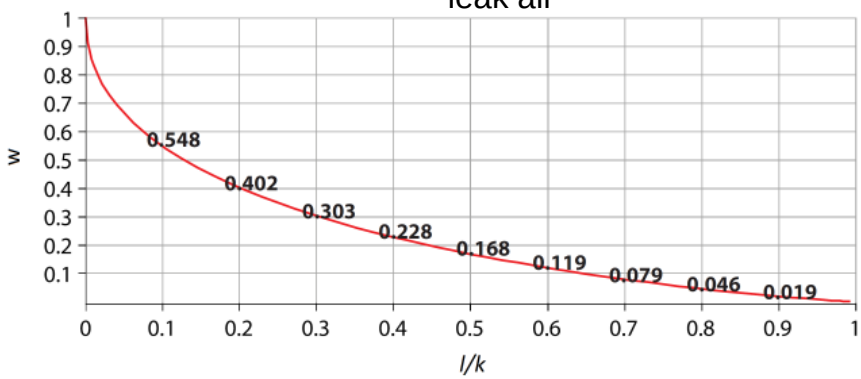
$$\text{Adv}(\mathcal{A}) \leq \frac{qt}{2^{p \cdot \text{w}(l/k)}}$$

- Where $\mathcal{A}$ has data complexity q, time complexity t, and w is the "number of bits of security" per probe

Big Key Cryptography (III)

- What is w(l/k) worth if $l/k = 0$?
- What is w(l/k) worth if $l/k = 1$?

leak all



probes for 128-bit sec., 1Gbit key, 100Mbit leaked? {...}

l/k = 100 M/ 1G = 0.1 -> w = 0.548

Summary

- Key exfiltration with low throughput can be mitigated
- $\exists$ efficient solutions for specific (symmetric) primitives
- But they force designers to re-think differently about how to protect emerging computer systems
- Up to now, focus (mostly) on software and security
- Next (last) threat: hardware Trojans and correctness
  - e.g., what if the manufacturer is an adversary?
    - & triggering events are rare (i.e., detection is hard)
  - e.g., how prevent denial-of-service attacks?

Hardware Trojans

- Ultimate attack where hardware is corrupted at manufacturing time (i.e., malicious manufacturer)
- Can be digital (i.e., trigger and payload via IOs) or physical (e.g., trigger and payload via other signals)
- Example of digital hardware hardware Trojan? $\{\dots\}$
- Example of physical hardware Trojan? $\{\dots\}$
- $\approx$ instantiation of the "device interception issue"

Reactive countermeasures

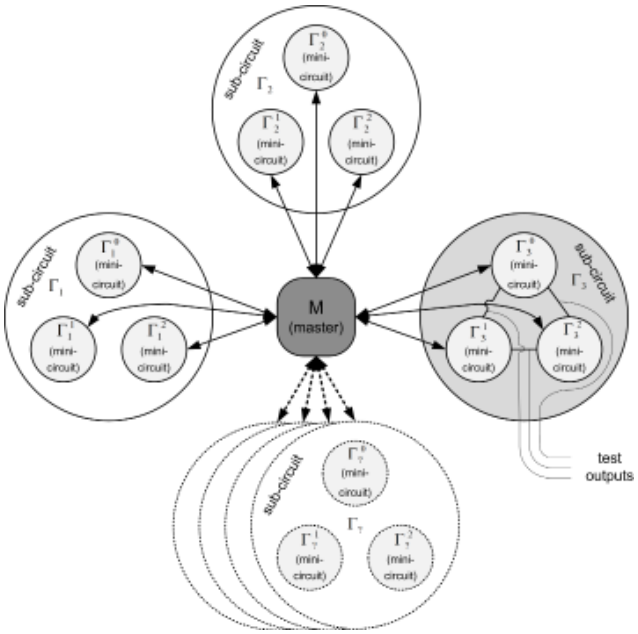
- Essentially based on detection: device inspection, reverse engineering, physical measurements, ...
- Conceptually unlimited but highly heuristic
  - Become unrealistic for large systems?

$\Rightarrow$ Minimizing the parts of the circuit to protect with reactive countermeasures is anyway (highly) desirable

Preventive countermeasures

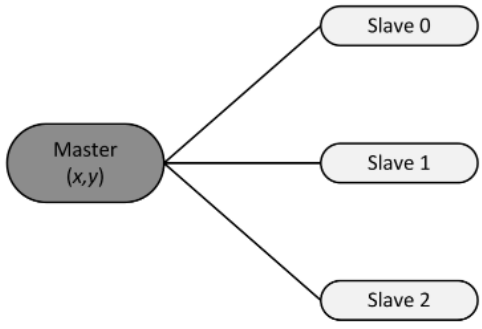
- Aim to make Trojan insertion/exploitation difficult
- e.g., *input scrambling* can be used to prevent digitally-triggered Trojans based on "cheat codes"
- How can we compute on scrambled inputs? MPC
- Next: how to (formally) prevent cheat codes & time bombs (i.e., digitally-triggered Trojans based on counters) and to guarantee correctness ($\neq$ security)

Trojan-resilience from MPC



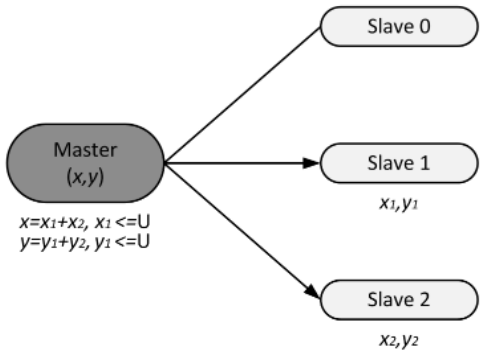
- Target Boolean circuit = XORs ($+$) and ANDs ($\cdot$)
- Trusted master
- 3-party computation on sub-circuits (vs. cheat codes)
- Redundancy & functional testing (vs. time bombs)

Initial setting



- Inputs (e.g., x, y) arrive via the (trusted) Master

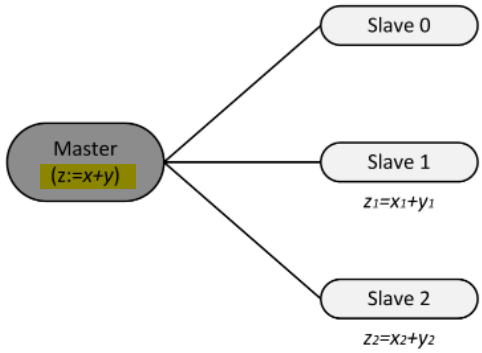
Input sharing



- Assumes Master has access to uniform randomness
  - Why does it prevent cheat codes? {...}

adversaryu cannot use x1,y1

Secure addition

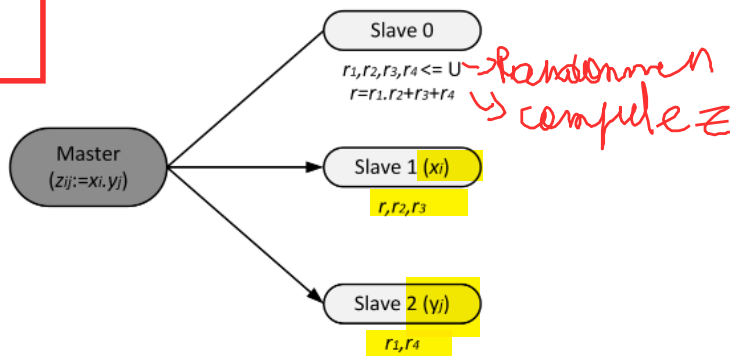


- Can be performed locally by slaves 1 & 2
  - Why is it functionally correct? {...}

Both share a portion of the sum. z1+z2 = all the rest

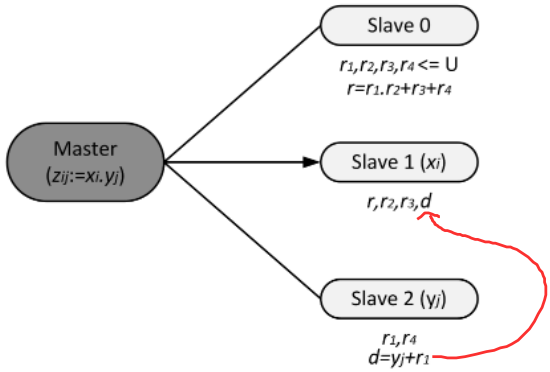
Shares multiplication (I)

-minimize : interaction, randomness and computation



- Slave 0 generates "correlated randomness"

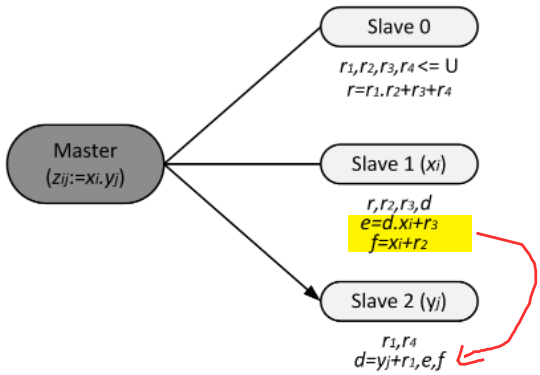
Shares multiplication (II)



- Slave 2 transmits a masked version of his share
  - Why is it secure (in semi-honest setting)? {...}

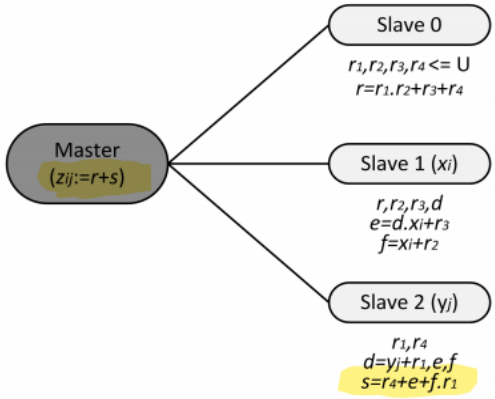
it doesn't leak r

Shares multiplication (III)



- Slave 1 transmits two masked versions of his share

Shares multiplication (IV)



XOR

- r and s are the two shares of the partial product
  - Why is it functionally correct? $\{\ldots\}$

Correctness: $r = r_4 + (y_j + r_1).x_i + r_3 + (x_i + r_2).r_1$

$= r_4 + x_i y_j + x_i r_1 + r_3 + x_i r_1 + r_1 r_2$

$= x_i y_j + r_1 r_2 + r_3 + r_4$

Secure multiplication

redondance

- Mini-circuit 1 computes $(r, s) = \text{MultShares}(x_1, y_2)$
- Mini-circuit 2 computes $(r', s') = \text{MultShares}(x_2, y_1)$
- Mini-circuit 1 computes $z_1 = x_1 \cdot y_1 + r + r'$
- Mini-circuit 2 computes $z_2 = x_2 \cdot y_2 + s + s'$
- Why is the randomness needed? {...}
  - Hint: think about a solution where $z_1 = x_1 \cdot y_1 + x_1 \cdot y_2$ and $z_2 = x_2 \cdot y_1 + x_2 \cdot y_2$

$y_1 + y_2 = y$ ↳ leaking!

$x_1(y_1 + y_2) = x_1 \cdot y \rightarrow$ AND $\rightarrow x_1 = 1 \Rightarrow y = 1$

Testing amplification

- So far MPC (roughly, see next) prevents cheat codes
- Time bombs can additionally be addressed as follows
  - Test every mini-circuit $t_r \xleftarrow{r} [0; t]$ times
    - Why randomizing? to desync bombs
  - Then run the Trojan-resilient circuit n times
  - Assume a Trojan-resilient circuit with λ sub-circuits
  - And let the Master compute the majority
  - Probability of a Trojan attack bounded by $\left(\frac{4m}{t}\right)^{\lambda/2}$ majorité
- Guarantees correctness for $n << t$ executions
- (Testing also needed to exploit semi-honest MPC)

Security vs. correctness

- Say you only want to produce pseudo-random numbers
- Then a (much) simpler solution is to
  - Run λ secure PRGs (without MPC)
  - XOR their output on the master (which remains secure as long as one PRG output is secure)
- Probability of a Trojan attack then bounded by $\left(\frac{4m}{t}\right)^{\lambda}$
- Useful for the secret sharing rand. (since never output)
- Keeps the Master simple(er) – i.e., XOR + MAJ gates

Summary

- Hardware Trojans adversaries are very (too?) powerful
- Wide classes of Trojans can be captured with MPC
  - But the cost of the MPC solution is high
- Others (physical) Trojans seem very hard to capture
- Interesting feature: MPC-based Trojan-resilient circuits do *not* assume non-collusion of the sub-circuits
- Better results could be obtained under this assumption
  - e.g., FPGAs from different vendors, ...
  - To capture more Trojans & to improve perfs.

Conclusions

- Snowden revelations $\approx$ a case for old (known) threats
- Shows crypto can be (and has been) circumvented
- One reason is probably the quest for performance
  - It is possible to do better (formally)
- Another (IMO, more worrying) reason is the lack of transparency in security engineering
- My view: proving security is a designer's task
- Open specs, hardware, software helps for that
  - e.g., in order to exploit formal methods/proofs

Privacy Enhancing Technologies

O. Pereira and F.-X. Standaert

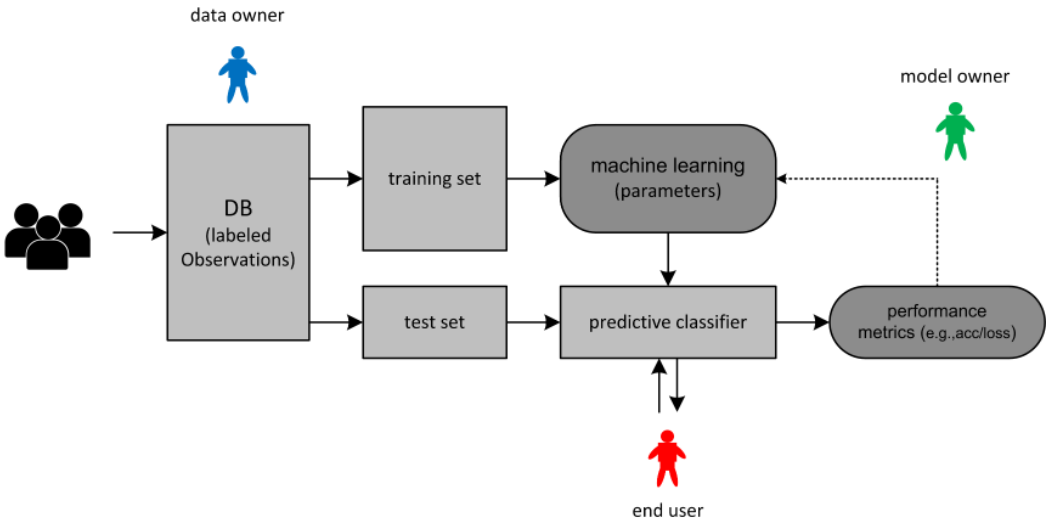
UCL Crypto Group, Université catholique de Louvain

Machine Learning Security and
Algorithmic Fairness (1 lecture)

Outline

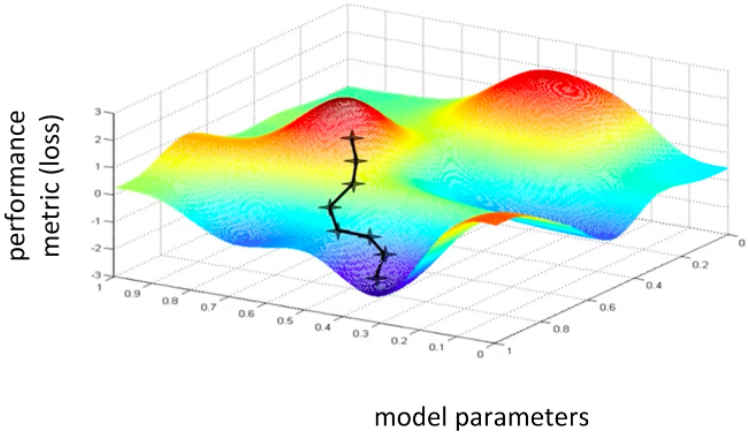
- Introduction / background
- Machine Learning Security
  - Adversarial inputs
  - Data poisoning
  - Model extraction/stealing
  - Membership inference
  - (Private learning)
- Algorithmic fairness
- Concluding remarks

Background: machine learning



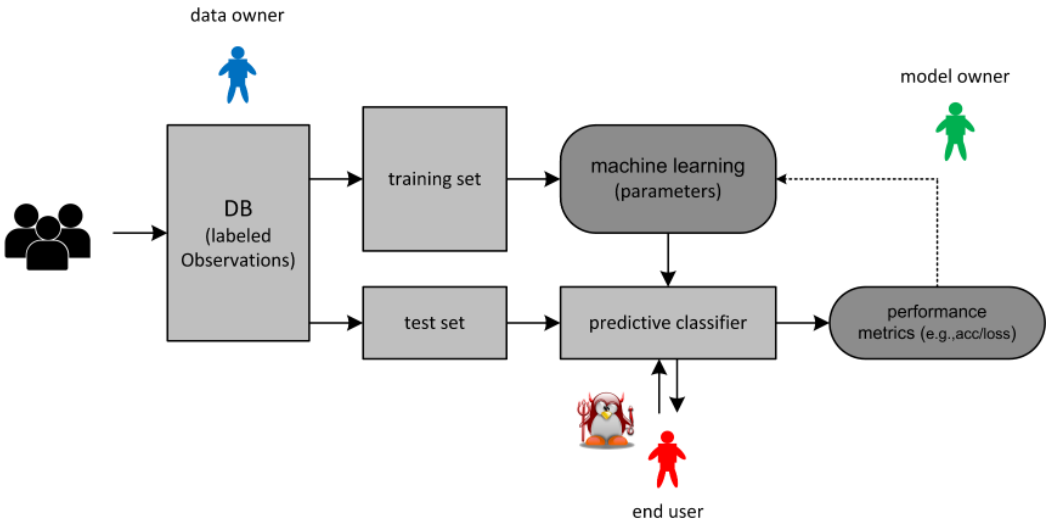
- Overly simplified! (see LELEC2870, LINGI2262, . . .)

Example: gradient descent



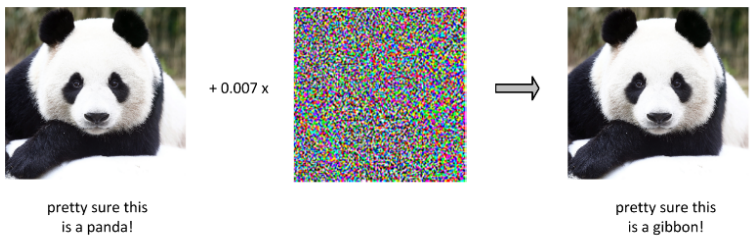
- ≈ random walk to find optimum (inherently heuristic)
  - With many parameters (step size, # of resets, . . .)

Adversarial examples



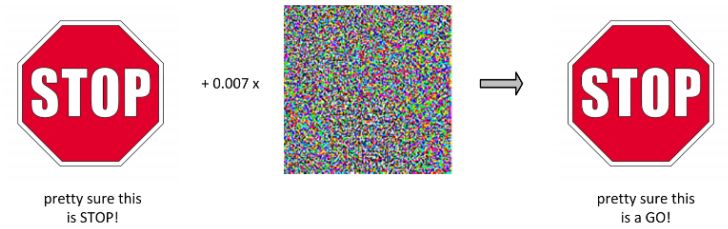
- Adv. crafts inputs to make the classifier fail

Illustration

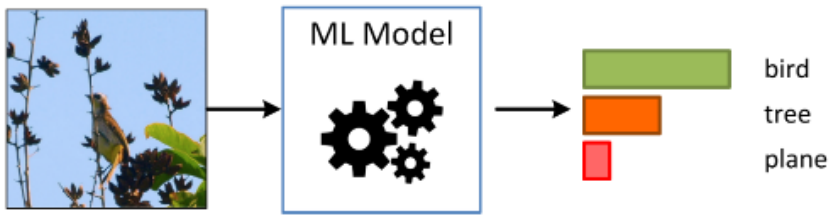


Illustration

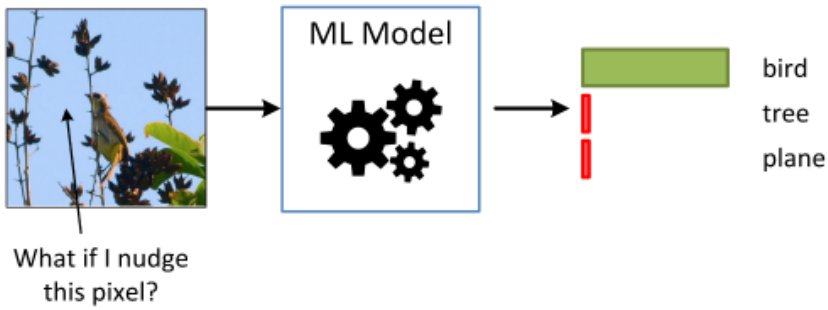
- Or (Though application to physical objects harder)



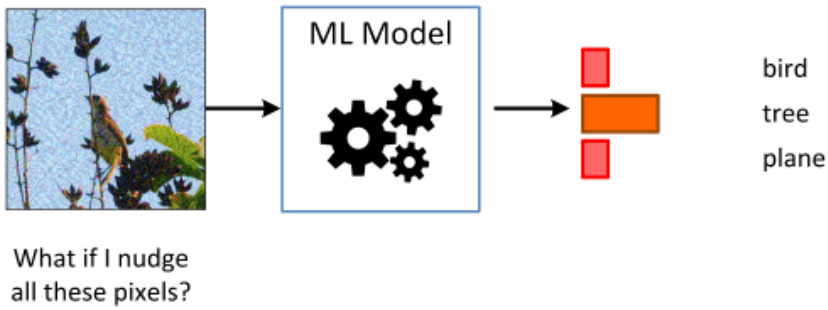
How does it work?



How does it work?



How does it work?

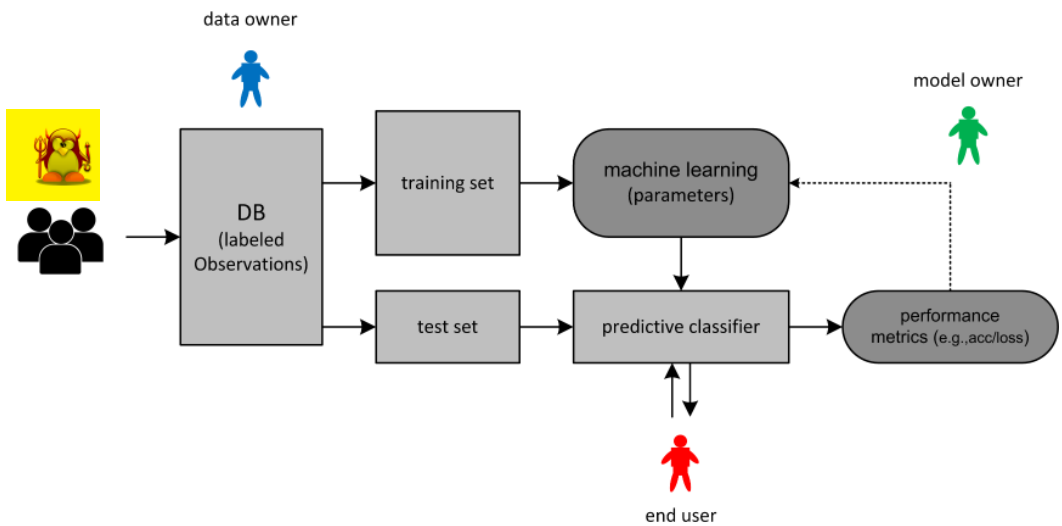


- Maximize model error with gradient *ascent*
- Possible in white box (i.e., the model is known)
- Also with black box access to the model {...}
  - Adv. ex. transfer/generalize across models

Countermeasures?

- Ensemble training (multi-models) ✗
- Preprocessing (blurring, . . .) ✗
- Detection of suspicious inputs ✗
- Adversarial training (heuristic but seems better)
  - Generate adv. ex. and re-inject in training set
  - Can be viewed as ≈ data augmentation in DL
- Improvements are easy / strong guarantees are hard
- Pr[misclassification] $< 2^{-80}$ ⫫ of adv. strategy?

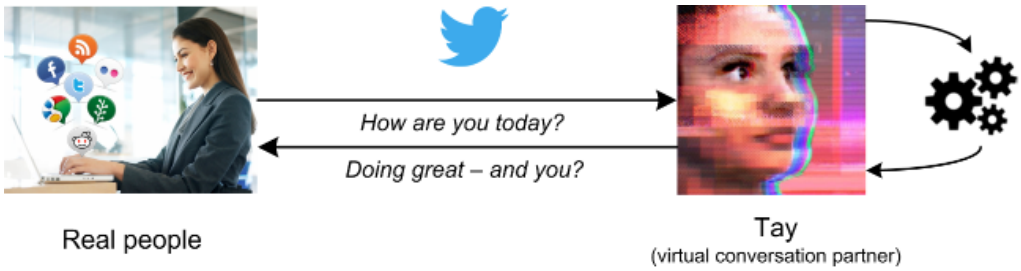
Data poisoning



- Adv. sends malicious data to bias the predictive model

Illustration

- Example from James Mickens' Usenix 2018 keynote



Illustration

- Example from James Mickens' Usenix 2018 keynote

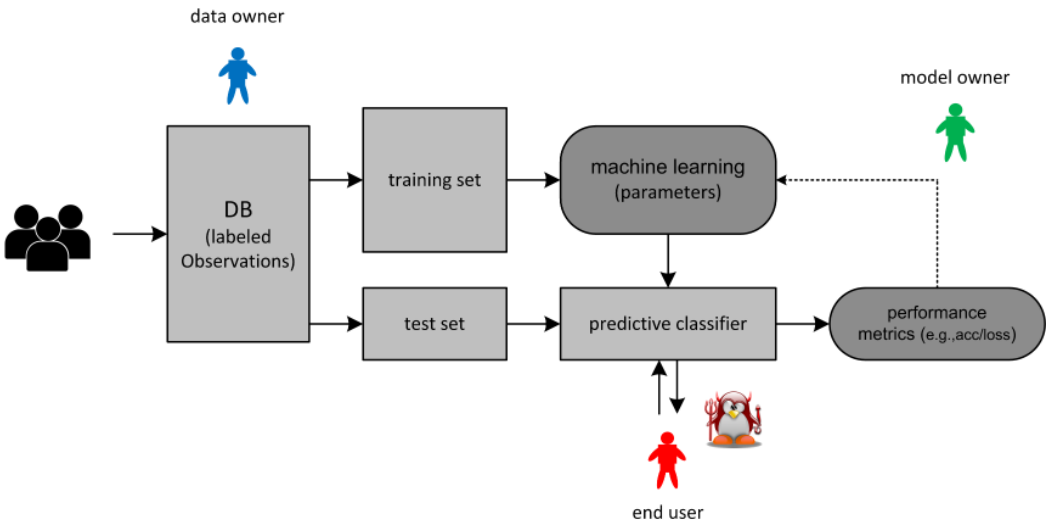
- Not even a technical problem (trivial to implement)
- But can become technical (e.g., malware detection)

Countermeasures?

- Robust statistics (e.g., outliers detection)
  - Possibly enables formal guarantees for some types of models (e.g., based on linear regression)
- Distributed data sources
  - For example, no single IP address should account for a too large fraction of the training set
- Compare newly trained models with former ones
- Golden dataset, data authentication, ...
- Strong & general guarantees again seem hard to reach

M. Jagielski et al., *Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning*, IEEE S&P 2018.

Model extraction, membership inference



1. Adv. tries to extrac the model from queries
2. Adv. tries to know whether a user's data $\in$ training set

Model extraction

- Why can it be simpler than model learning? no need of data (hard to gather)

	learning f(x)	extracting f(x)
function	complex / noisy	simpler / deterministic
labels	hard	hard or soft
labeling	by humans	on-query x

- Logistic regression needs $|\text{data}| \approx |\text{features}| \cdot 10$
  But its model extraction costs $|\text{data}| \approx |\text{features}| + c$
- Large neural nets / decision trees less understood

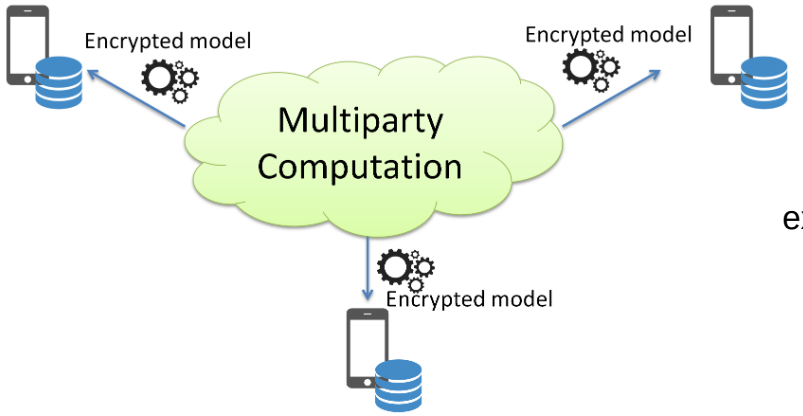
S. Milli et al., *Model Reconstruction from Model Explanation*, ACM FAT 2019.

Membership inference

- Generally target "Machine Learning as a service"
- Typical examples of applications
  - Learn whether a patient's data is in a DB
  - Learn whether a customer has a competing account
    - Netflix vs. Amazon, Apple vs. Spotify, ...
- Closely related to overfitting (model's behavior must be significantly different on training set and test set)
- Countermeasures?
  - Differential privacy (but utility cost)
  - Well estimated models (& outliers detection)

R. Shokri et al., *Membership Inference Attacks against ML Models*, IEEE S&P 2017.

Private learning



expensive

- See previous lectures on Big Data Privacy

Y. Lindell, B. Pinkas, *Privacy-Preserving Data Mining*, JoC, 2002.

Algorithmic fairness (non-tech.)

- Assume machine learning security is solved
- The next question is: is machine learning fair?
- So what do we mean by fair (non-discriminating)?

Algorithmic fairness (non-tech.)

- So what do we mean by fair (non-discriminating)?
  - Human discrimination sometimes requires intent
  - Intent is hard to define for algorithms (intentions)
    - Unless they are designed with intentional bias
    - Algorithmic transparency is a problem of its own
    - And devices are hard to trust (malware, HW Trojans)
  - So broader definitions need to be considered
    - E.g., need to treat people as individuals (not groups)

R.H. Sloan, R. Warner, *When is an Algorithm Transparent: Predictive Analytics, Privacy and Public Policy* / R. Binns, *What Can Political Philosophy Teach Us about Algorithmic Fairness*, IEEE Security & Privacy Magazine, May/June 2018.

Algorithmic fairness (tech.)

- Fairness criteria are also hard to define technically
- Consider a target variable Y (e.g., interviewed people), attribute A (e.g., gender) and classifier score R
  - Note: it is hard to exhaust groups / attributes
- One requirement could be independence: $R \perp\!\!\!\perp A$
  - Yet not always desirable $\{\dots\}$
- Another requirement could be separation: $R \perp\!\!\!\perp A|Y$
- Technical caveat – incompatible notions:

Lemma. *Assume Y is binary, A is not $\perp\!\!\!\perp$ of Y and R is not $\perp\!\!\!\perp$ of Y, then independence & separation cannot both hold*

Summary

- Ultimately, data is always biased (and completely removing biases implies completely canceling utility)
- As the size of datasets increases, the risk of spurious correlations also increases significantly
  - What is a spurious correlations? $\{\dots\}$

$\Rightarrow$ automated data processing must be considered with care

- One legal solution: right to explanation (reading)

C.D. Calude, G. Longo, *The Deluge of Spurious Correlations in Big Data*, Foundations of Science, vol 22, issue 3, 2017.

Conclusions

- Privacy was already difficult to define
- Machine learning security is worse
- Algorithmic fairness is even worse

- Machine learning is useful to dig into big data
- Application to security (e.g., malware detection), which usually requires worst-case, is challenging
- Automation create a lot of fairness issues
  - Data can easily reproduce existing biases
- Summary: be critical (using technology is not neutral)