UNIVERSITY OF TWENTE.

Security Building Blocks

Specical thanks to Luan Ibraimi

OUTLINE

- Algebra recap
- Cryptography recap
- Commitment Schemes
- Secret Sharing
- Functional Encryption
- Elliptic curves

ASSUMED TO BE KNOWN…

If you do not know any of these terms, get yourself familiar with it a.s.a.p.!

- Symmetric key encryption
- Public key encryption
- Hash functions, cryptographic hash functions
- Random numbers: true random numbers vs. numbers that are computationally indistinguishable from random
- Message Authentication Codes (MACs)
- Basics of linear algebra, modular arithmetic
- Groups (algebraic structure)
- basics of projective space

If you think you don’t have sufficient knowledge in cryptography:

Nigel Smart - Cryptography: An Introduction

Katz, Lindell -- Introduction to Modern Cryptography

Algebra recap

(you should already be familiar with most of it)

AN ALGEBRAIC STRUCTURE: GROUP

- Group: suppose we have any binary operation, such as multiplication (·), that is defined for every pair of elements in a set G, which is denoted as (G, ·)
- Then *G* is a *group* with respect to multiplication if the following conditions hold:

1.) ***G* is *closed* under multiplication**: $x \in G$, $y \in G$, imply $x \cdot y \in G$

2.) **· is associative**. For all x, y, z, $\in G$, $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

3.) ***G* has an identity element *e***. There is an *e* in *G* such that $x \cdot e = e \cdot x = x$ for all $x \in G$.

4.) ***G* contains inverses**. For each $x \in G$, there exists $y \in G$, such that $x \cdot y = y \cdot x = e$.

- Instead of multiplication one can also use addition (+), or another operation

IMPORTANT DEFINITIONS RELATED TO GROUPS (1/2)

- **Generator**: g is a generator of a group G if $G = \{g^1, g^2, \ldots, g^o\}$ where o is the order of the group

- **Cyclic group**: A group G is cyclic if it can be generated by one generator g

- **Multiplicative group of integers modulo $\boldsymbol{n}$**: its elements are the primitive residue classes modulo n (i.e. the numbers between 1 and n that are relatively prime to n). The operation is multiplication $mod\ n$.
  - E.g.: $Z_9^* = \{1,2,4,5,7,8\}$ where $4 \cdot 8 = 5$ (because $32 \equiv 5\ mod\ 9$)
- **Additive group of integers modulo $\boldsymbol{n}$**: the integers from 0 to $n-1$. The operation is addition $mod\ n$.
  - E.g.: $Z_8 = \{0,1,2,3,4,5,6,7\}$ where $5 + 6 = 3$ (because $11 \equiv 3\ mod\ 8$)

IMPORTANT DEFINITIONS RELATED TO GROUPS (2/2)

- **Commutative group:** a group $(G,\odot)$ is commutative iff $\forall a, b \in G$:

$$a \odot b = b \odot a$$

  - Not every group is commutative!

ANOTHER ALGEBRAIC STRUCTURE: A FIELD

A field $\mathbb{F}$ is:

- a set of elements S $(\mathbb{N}, \mathbb{Z}, \mathbb{R}, \ldots)$
- two operators, typically $+$ and $\cdot$

such that

- if $a, b \in S$ and $\odot \in \{+, \cdot\} \implies a \odot b \in S$ (closure)
- $a + (b + c) = (a + b) + c$ (associative)
- $a + b = b + a$ (commutative)
- $\exists 0 : a + 0 = a$ and $\exists 1 : 1 \cdot a = a = a \cdot 1$ (additive and multiplicative identity)
- $\forall a \in S : \exists -a : a - a = 0$ (additive inverse)
- $\forall a \in S : \exists a^{-1} : a \cdot a^{-1} = 1$ (multiplicative inverse)
- $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ (distributivity)

·	O	I	A	B
O	O	O	O	O
I	O	I	A	B
A	O	A	B	I
B	O	B	I	A

+	O	I	A	B
O	O	I	A	B
I	I	O	B	A
A	A	B	O	I
B	B	A	I	O

Cryptography recap

(you should already be familiar with most of it)

CRYPTOSYSTEMS AND RANDOMNESS

- Deterministic Encryption: encrypting the *same plaintext* with the *same key* will always yield the *same ciphertext*
  - What is the main weakness of this kind of encryption? Is it always a weakness?
- Probabilistic Encryption: encrypting the *same plaintext* with the *same key* will result in a *different ciphertexts*
  - randomness is used in the encryption

ENCRYPTION: WHO TO WHO?

- Symmetric Key Encryption (SKE): one party encrypts, one decrypts (and the other way around as well → symmetric)
- Public Key Encryption (PKE): many encrypts, one decrypts (not symmetric)
- Broadcast Encryption (BE): one encrypts, many decrypts (not symmetric)

- key encapsulation: encrypting the message with SKE and encrypting the symmetric key using PKE
  - PKE is generally much less efficient than SKE, so when a PKE scheme is needed, often key encapsulation is the best option

SECURITY OF CRYPTOSYSTEMS: TRUST

When designing a cryptosystem, we can assume different levels of trustworthiness of the different parties in the system:

- **Honest party**: it follows the protocol and does not do anything else
- **Honest-but-curious (HBC, semi-honest, semi-trusted) party**: it follows the protocol but tries to learn as much as possible (about the other parties' secrets)
- **malicious party**: it does not follow the protocol. Its goal can be various, e.g. learning as much as possible, misleading other parties, sabotaging the system etc.
- **Untrusted party**: no unanimous definition, typically (but not exclusively!) used for HBC parties (when the authors try to overplay the security features of their work…)
- **collusion**: when different actors share (parts of) their knowledge and possibly their computational power to learn more (about the system, about other parties etc.)

MODELS FOR SECURITY PROOFS

- Random Oracle: a theoretic black box that outputs a true random number for any given input, and have the same output for the same input.

Most cryptographic schemes are proven secure in one of these 3 models:

- **Random Oracle Model (ROM):** we treat the cryptographic hash functions in the scheme as random oracles. The adversary can query these random oracles.
- **Generic Group Model (GGM):** the adversary only knows a random encoding of the group(s) used in the scheme and not an efficient one. Thus, the adversary has to query an oracle to perform a group operation (or a pairing operation, if it is a bilinear group)
- **Standard Model (STM):** the adversary has limited computational power and limited time to break the proposed scheme.

SECURITY ANALYSIS

- Any proposed cryptographic scheme needs to have a proper **security analysis**:
  - a mathematical guarantee that a scheme cannot be broken by a certain class of attackers

REQUIREMENTS OF A SECURITY ANALYSIS

- A security analysis needs to provide:
  - **A precise description of the scheme**: the participants, their roles, the amount of trust we have in them, the algorithms they run, and the communication between them
  - **A precise description of the class of attackers**: computational power, available time, role in the protocol, ability to corrupt participants (collusion), the extent to which they follow the protocol
  - **A precise description of the model**
  - **A precise description of the assumptions**: certain mathematical problems are assumed to be very hard to compute
  - **A precise description of the “win condition”**: when the security of the scheme is considered broken
  - **A proof** that no attacker can achieve the win condition for the proposed scheme in the given model with the described assumptions

SECURITY ANALYSIS COMPLEMENT

- Aspects that have to be defined for a **class of attackers**:
  - deterministic vs. probabilistic
  - polynomial time vs. exponential time vs. unlimited time
  - computationally bounded vs. comp unbounded
  - colluding vs. not colluding
  - server vs. user vs. third party vs. else
    - if it plays a role in the scheme: semi-honest vs. malicious
  - adaptive vs. non-adaptive

THE SETTING OF SEARCHABLE ENCRYPTION

FOR SEARCHABLE ENCRYPTION

Searchable Encryption algorithms:

`Keygen`(k): Outputs: master secret key msk and public parameters $param$

`Encrypt`$(param, W, M)$: Outputs a ciphertext $S_{W,M}$

`Trapdoor`(W', msk): Outputs a trapdoor $T_{W'}$

`Decrypt`$(T_{W'}, S_{W,M})$: Outputs M iff $W = W'$

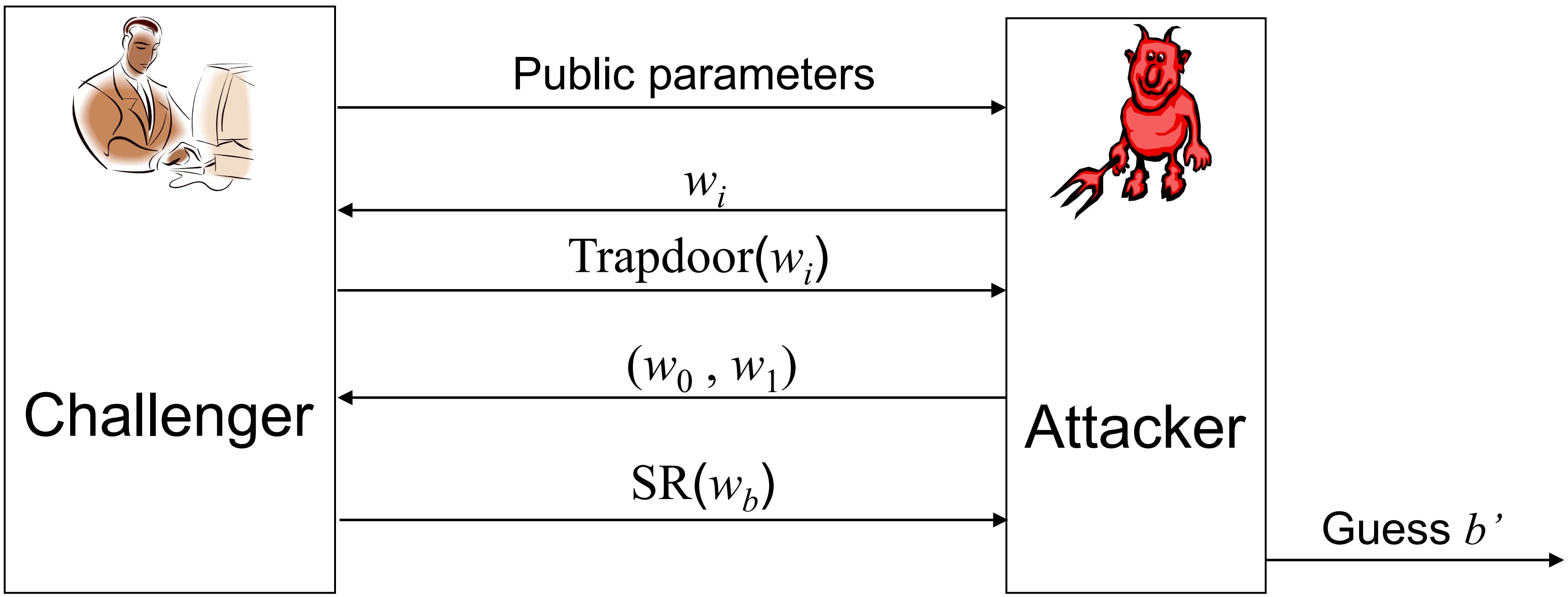
Security requirement: M and W must be hidden.

A provably secure scheme must show that from perspective of a *polynomially bounded* adversary:

1- Ciphertext is indistinguishable from random

2- Trapdoor of other keywords do not reveal information on ciphertext

ATTACKER MODEL IN SEARCHABLE ENCRYPTION



$$\Pr[b = b'] = ½ + \varepsilon$$

If ε is not negligible attacker wins game and scheme is not secure

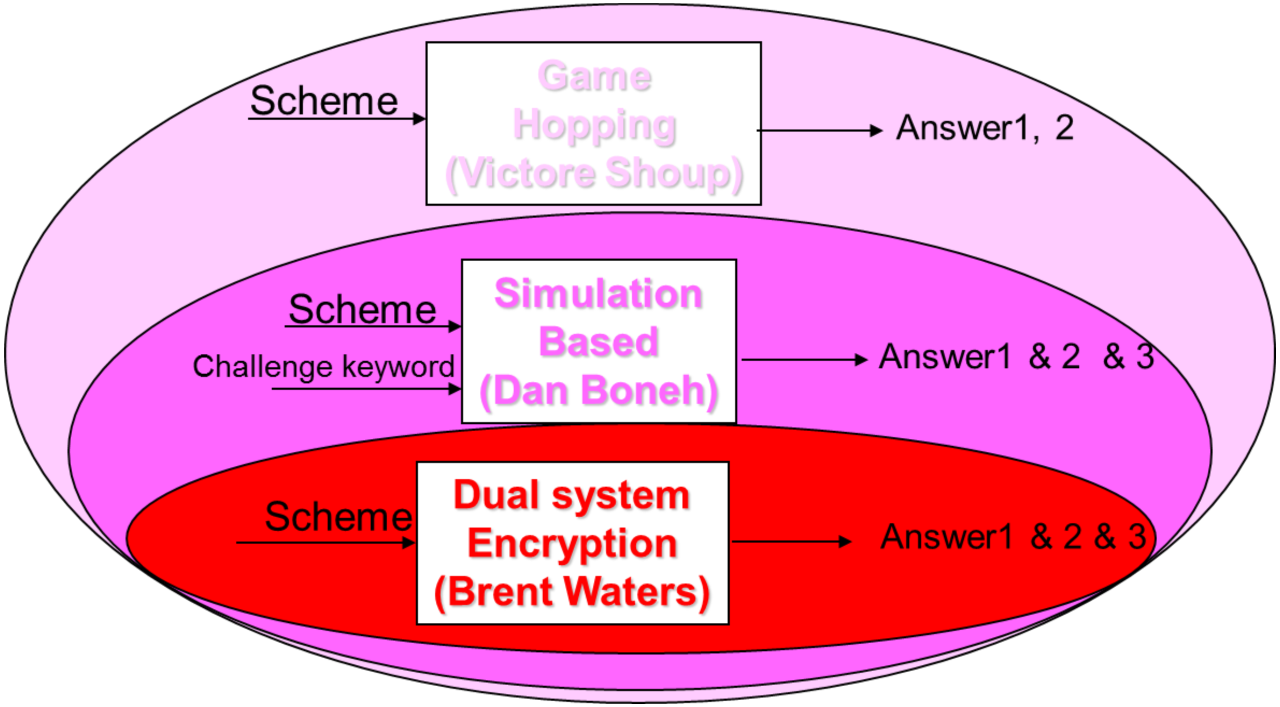
SECURITY PROOF METHODOLOGY FOR SEARCHABLE ENCRYPTION

To prove that is ε negligible security proof must answer these questions

1- Is ciphertext indistinguishable from random?

2- Does trapdoor of keywords other than challenge keyword reveal information on challenge ciphertext?

3- Is trapdoor of challenge keyword simulateble



A SECURITY PROOF TECHNIQUE: GAME HOPPING

SECURITY GAMES

- A series of games is defined.
- Game 1: a game between an attacker and a challenger in IND-CPA* model to break proposed crypto scheme.
- Let S_1 be event that attacker wins game,
- $\Pr[S_1] = ½ + \varepsilon$

- Game 2: a game between an attacker and a challenger in IND-CPA model to break a crypto scheme which is information theoretically secure.
- Let S_2 be event that attacker wins game,
- $\Pr[S_2] = ½$

- $\Pr[S_1] - \Pr[S_2] = \varepsilon$
- If we can show that $\Pr[S_1] - \Pr[S_2]$ is negligible then ε is negligible

- * IND-CPA: INDistinguishable under Chosen Plaintext Attack

A SECURITY PROOF TECHNIQUE: GAME HOPPING

THE GOAL

- To prove that $Pr[S_1]$ - $Pr[S_2]$ is negligible a distinguisher algorithm, and two distributions P_1 and P_2 are used such that:

  - By assumption:
    Pr[Dist. Outputs 1 from P_1] – Pr[Dist. Outputs 1 from P_2]
    is negligible.

  - Pr[Dist. Outputs 1 from P_1] = $Pr[S_1]$
  - Pr[Dist. Outputs 1 from P_2] = $Pr[S_2]$

  Then we can prove that:
- $Pr[S_1] – Pr[S_2]$ is negligble

A SECURITY PROOF TECHNIQUE: GAME HOPPING

AN EXAMPLE PROOF: ELGAMAL SCHEME

- **`Keygen`**(s): Pick a random y
  - Master secret key: $msk = y$
  - Public parameters: $pk = g^y$
- **`Encrypt`**(m , pk) :Pick a random x:

$$Enc_m = (g^x , mg^{xy})$$

- **`Decrypt`**(Enc_m , msk): Let $Enc_m = (a , b)$

$$m = b/a^y$$

EXAMPLE PROOF: ELGAMAL SCHEME

ASSUMPTIONS

- Let B be an algorithm that given a tuple $(g^{z1}, g^{z2}, \ldots, g^{zl}, Z)$
  - Outputs 1 if Z is a function of $z_1, z_2, .., z_l$
  - Outputs 0 if Z is a random

- Decision Diffie-Hellman (DDH) assumption
  - Informally: Given (g^{z_1}, g^{z_2}) it is hard to distinguish between $g^{z_1 z_2}$ and a random Z.
  - Formally: $|\Pr[B(g^{z_1}, g^{z_2}, g^{z_1 z_2}) = 1] - \Pr[B(g^{z_1}, g^{z_2}, Z) = 1]| < \varepsilon_{DDH}$
  - ε_{DDH} is negligible

GAME1 IN THE ELGAMAL PROOF

Challenger | Attacker

Setup

$$\xrightarrow{g^y}$$

Query

$$\xleftarrow{m_i}$$

$$\xrightarrow{Enc_{mi} = (g^{x_i}\ ,\ m_i\, g^{x_i y})}$$

Challenge

$$\xleftarrow{(m_0\ ,\ m_1)}$$

Guess

$$\xrightarrow{Enc_{mb} = (g^{x}\ ,\ m_b\, g^{xy})}$$

Guess b'

$$\Pr[b = b'] = ½ + \varepsilon$$

GAME 2 IN THE ELGAMAL PROOF

Challenger | Attacker

Setup

$$\text{Challenger} \xrightarrow{g^y} \text{Attacker}$$

Query

$$\text{Challenger} \xleftarrow{m_i} \text{Attacker}$$

$$\text{Challenger} \xrightarrow{Enc_{mi} = (g^x\ ,\ m_i\, g^{xy})} \text{Attacker}$$

Challenge

$$\text{Challenger} \xleftarrow{(m_0\ ,\ m_1)} \text{Attacker}$$

Guess

$$\text{Challenger} \xrightarrow{Enc_{mc} = (g^x\ ,\ m_c\, g^z)} \text{Attacker}$$

Guess c'

$$\Pr[c = c'] = ½$$

ALGEBRAIC STRUCTURES BEHIND THE ENCRYPTION SCHEMES

- Both the plaintext messages and the ciphertexts are elements of an algebraic structure
  - the *message space* and the *ciphertext space* are usually the same, but not always
  - some widely used examples:
    - integers modulo n
    - Finite fields (Galois fields)
    - cyclic groups of prime order
    - **elliptic curve groups**

Commitment Scheme

COMMITMENT SCHEME

- Suppose Alice and Bob want to flip a coin to decide something.
  - However, they are not physically in the same place.
  - How can they flip a coin over the phone?
  - If Alice flips the coin, she might want to manipulate the result so that it is to her favor.
  - If Bob flips the coin, he might do the same thing.

COMMITMENT SCHEME

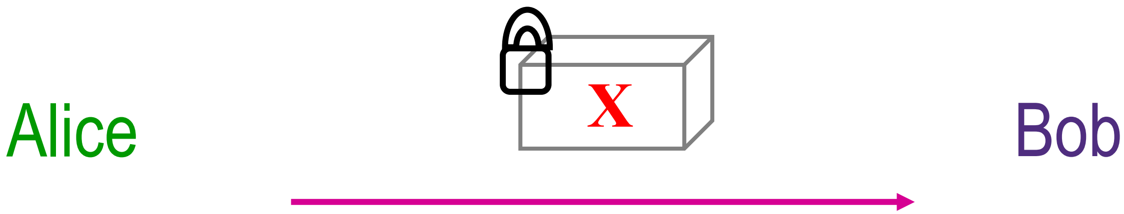
- One possible solution is:
  - Alice flips a coin and commits to it.
  - Bob flips another coin and tells Alice his result.
  - Alice reveals her own result what she committed to
  - If Alice’s revelation matches the coin result Bob reported, Alice wins.
- But how can Alice commit?

COMMITMENT SCHEME

- A commitment scheme allows Alice to compute a commitment, such that:
  - Alice can reveal the value later.
  - Alice cannot cheat (i. e., give a false value) when revealing the value.
  - Bob can verify the committed value.

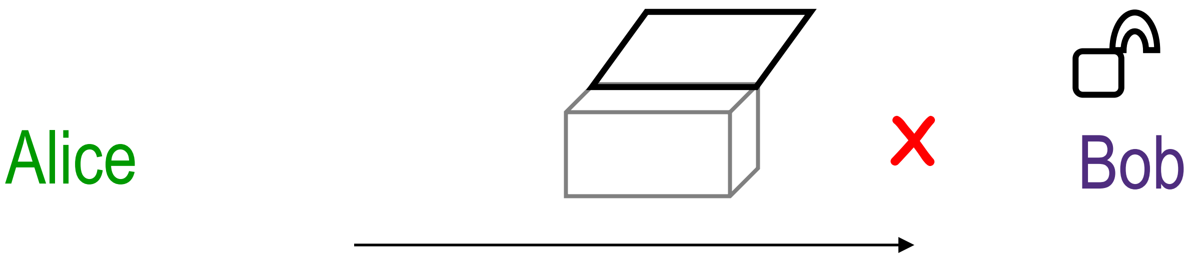
COMMITMENT SCHEME

Commit Phase



Alice is bound to x

Reveal Phase



Bob can verify x was the value in the box

SECURITY PROPERTIES OF A COMMITMENT SCHEME

- Hiding

- at the end of Commit phase, no adversarial receiver learns information about the committed value

- Binding

– at the end of Reveal phase, no adversarial sender can successfully reveal two different values

COMMITMENT SCHEMES: FORMAL SECURITY PROPERTIES

- **Two kinds of adversaries**
  - those with infinite computation power and those with limited computation power
- **Unconditional hiding**
  - the commitment phase does not leak any information about the committed message, in the information theoretical sense (similar to perfect secrecy)
- **Computational hiding**
  - an adversary with limited computation power cannot learn anything about the committed message (similar to semantic security)

COMMITMENT SCHEMES: FORMAL SECURITY PROPERTIES

- **Unconditional binding**
  - after the commitment phase, an infinite powerful adversary sender cannot reveal two different values
- **Computational binding**
  - after the commitment phase, an adversary with limited computation power cannot reveal two different values

GENERAL COMMITMENT: PEDERSEN SCHEME (1/2)

- **Example Scheme (by Pedersen):**
  - Let g and $h=g^a$ be two generators mod large prime p, picked independently.
  - Commitment to x: $c=g^x h^r$, where r is a random number.
  - To open the commitment the sender sends x and r.
  - The receiver verifies whether $c=g^x h^r$

GENERAL COMMITMENT: PEDERSEN SCHEME (2/2)

- **Unconditionally hiding**
  - Given a commitment c, every value x is equally likely to be the value committed in c.
  - For example, given x,r, and any x', there exists r' such that $g^x h^r = g^{x'} h^{r'}$, in fact $r' = (x-x')a^{-1} + r \bmod q$.

- **Computationally binding**
  - Suppose the sender open another value $x' \neq x$. That is, the sender find x' and r' such that $c = g^{x'} h^{r'} \bmod p$. Now the sender knows x,r,x', and r' s.t., $g^x h^r = g^{x'} h^{r'} \pmod p$, the sender can compute $a = (x'-x)\cdot(r-r')^{-1}$. Assume DL is hard, the sender cannot open the commitment with another value.

Secret Sharing

SECRET SHARING

- Suppose a company has a very important secret. Who should know this secret?
  - If only the CEO knows it, then what if something unexpected happened to him?
  - If a good number of people (e.g., all directors) know it, then what if one of them were corrupted?
  - A cryptographic solution to this problem is secret sharing.

SECRET SHARING

*‘**Secret sharing** (also called **secret splitting**) refers to method for distributing a secret amongst a group of participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number, of possibly different types, of shares are combined together; individual shares are of no use on their own’*

[source: Wikipedia]

SECRET SHARING
FORMAL DEFINITION

More formally, a (t, n)-threshold secret sharing scheme is a scheme, where

- a secret $s = s_0$ is shared with
- n parties, where
- party i ($i \in \{1, \ldots, n\}$) receives a share s_i, such that
- you need at least t parties to reconstruct s

SECRET SHARING
A TRIVIAL SOLUTION

For $t = n$ there is a trivial solution:

1. Encode the secret to an integer s
2. Generate $n - 1$ random values: $s_1, \ldots, s_{n-1}$
3. Calculate
$$s_n = s - \sum_{i=1}^{n-1} s_i$$
4. Give each party p_i the value s_i

SECRET SHARING

SOLUTION FOR t<n

In general, for a (t, n)-threshold Shamir Secret Sharing scheme you need:

1. Choose a random polynomial $f(x)$ of degree $t-1$,
2. such that $f(0) = s$
3. Compute n points $(i, f(i))$ with $(i \neq 0)$
4. distribute the points over the parties

SHAMIR SECRET SHARING

Preparation

Suppose that our secret is 1234 $(S = 1234)$.

We wish to divide the secret into 6 parts $(n = 6)$, where any subset of 3 parts $(k = 3)$ is sufficient to reconstruct the secret. At random we obtain 2 numbers: 166, 94.

$$(a_1 = 166; a_2 = 94)$$

Our polynomial to produce secret shares (points) is therefore:

$$f(x) = 1234 + 166x + 94x^2$$

We construct 6 points from the polynomial:

$$(1, 1494); (2, 1942); (3, 2578); (4, 3402); (5, 4414); (6, 5614)$$

We give each participant a different single point (both x and $f(x)$).

SHAMIR SECRET SHARING

In general, for a (t, n)-threshold Shamir Secret Sharing scheme you need:

1. Choose a random polynomial $f(x)$ of degree $t-1$,
2. such that $f(0) = s$
3. Compute n points $(i, f(i))$ with $(i \neq 0)$
4. distribute the points over the parties

To recover the secret

1. take any t points and
2. use Lagrange interpolation to reconstruct $f(x)$
3. calculate the secret $s = f(0)$

SHAMIR SECRET SHARING

RECONSTRUCTION: THE FORMULA

$$\ell_j(x) := \prod_{\substack{0\le m\le k\\ m\ne j}} \frac{x-x_m}{x_j-x_m} = \frac{(x-x_0)}{(x_j-x_0)} \cdots \frac{(x-x_{j-1})}{(x_j-x_{j-1})} \frac{(x-x_{j+1})}{(x_j-x_{j+1})} \cdots \frac{(x-x_k)}{(x_j-x_k)}.$$

$$\ell_j(x) := \prod_{\substack{0\le m\le k\\ m\neq j}} \frac{x-x_m}{x_j-x_m} = \frac{(x-x_0)}{(x_j-x_0)}\cdots\frac{(x-x_{j-1})}{(x_j-x_{j-1})}\frac{(x-x_{j+1})}{(x_j-x_{j+1})}\cdots\frac{(x-x_k)}{(x_j-x_k)}.$$

SHAMIR SECRET SHARING

RECONSTRUCTION

Reconstruction

In order to reconstruct the secret any 3 points will be enough.

Let us consider $(x_0, y_0) = (2, 1942)\,;(x_1, y_1) = (4, 3402)\,;(x_2, y_2) = (5, 4414)$.

We will compute Lagrange basis polynomials:

$$\ell_0 = \frac{x-x_1}{x_0-x_1}\cdot\frac{x-x_2}{x_0-x_2} = \frac{x-4}{2-4}\cdot\frac{x-5}{2-5} = \frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}$$

$$\ell_1 = \frac{x-x_0}{x_1-x_0}\cdot\frac{x-x_2}{x_1-x_2} = \frac{x-2}{4-2}\cdot\frac{x-5}{4-5} = -\frac{1}{2}x^2 + \frac{7}{2}x - 5$$

$$\ell_2 = \frac{x-x_0}{x_2-x_0}\cdot\frac{x-x_1}{x_2-x_1} = \frac{x-2}{5-2}\cdot\frac{x-4}{5-4} = \frac{1}{3}x^2 - 2x + \frac{8}{3}$$

Therefore

$$f(x) = \sum_{j=0}^{2} y_j \cdot \ell_j(x)$$

$$= 1942\cdot\left(\frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}\right) + 3402\cdot\left(-\frac{1}{2}x^2 + \frac{7}{2}x - 5\right) + 4414\cdot\left(\frac{1}{3}x^2 - 2x + \frac{8}{3}\right)$$

$$= 1234 + 166x + 94x^2$$

Recall that the secret is the free coefficient, which means that $S = 1234$, and we are done.

SHAMIR SECRET SHARING

In general, for a (t, n)-threshold Shamir Secret Sharing scheme you need:

1. Choose a random polynomial $f(x)$ of degree $t - 1$,
2. such that $f(0) = s$
3. Compute n points $(i, f(i))$ with $(i \neq 0)$
4. distribute the points over the parties

To recover the secret

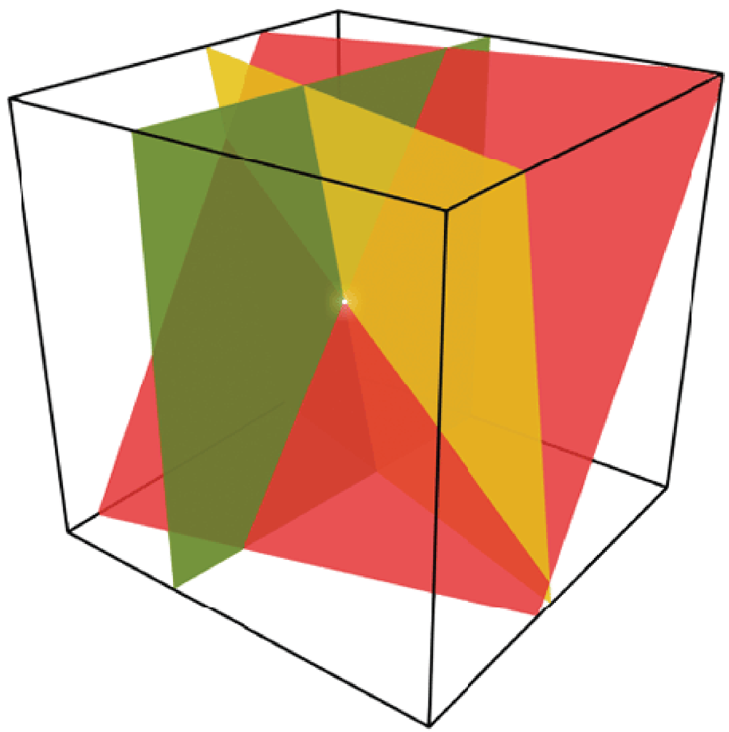
1. take any t points and
2. use Lagrange interpolation to reconstruct $f(x)$
3. calculate the secret $s = f(0)$

BLAKLEY SECRET SHARING

- Another early secret sharing scheme:
  - Map s to a point in t-dimensional space.
  - Choose n random (t-1)-dimensional hyperplanes that contain s.
  - Each hyperplane is a share.
  - To recover s only needs to compute the intersection of t hyperplanes.
  - Having <t shares cannot tell what is s.

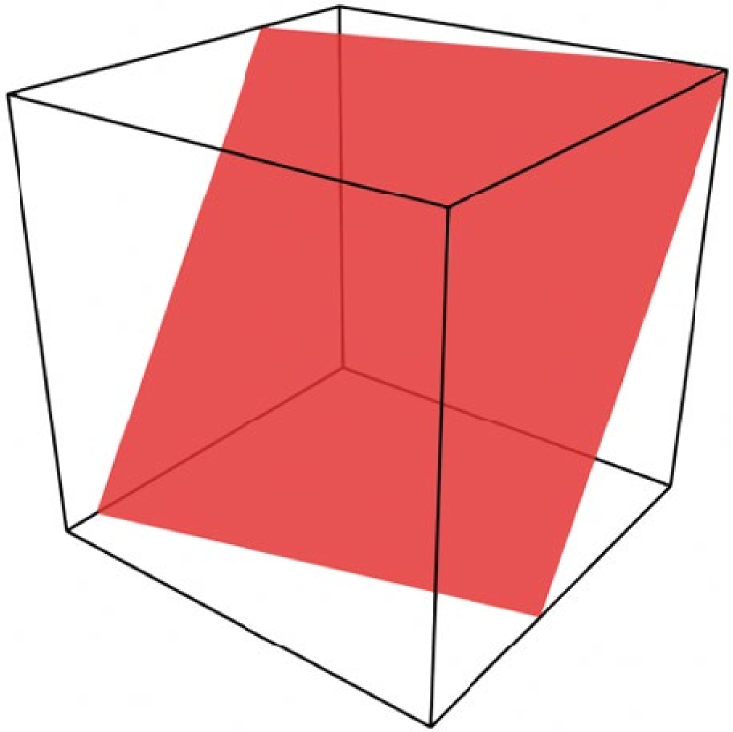
BLAKLEY SECRET SHARING

Secret is the intersection point



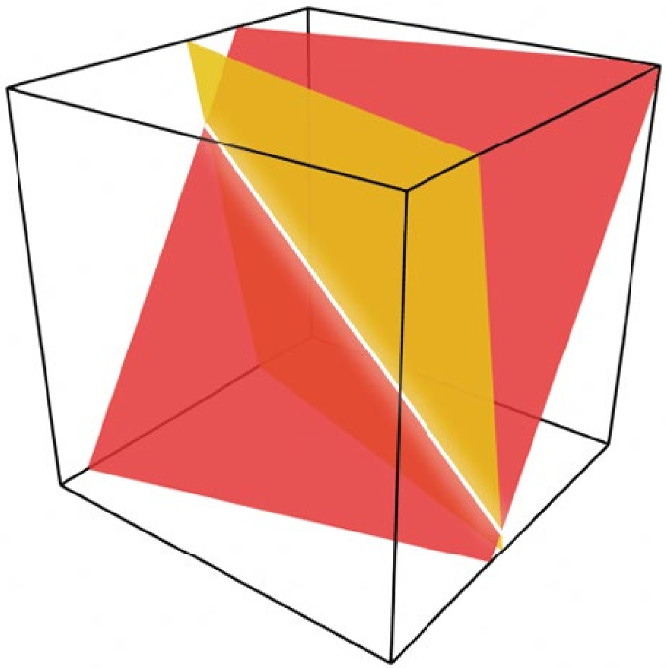
BLAKLEY SECRET SHARING

Each party only receives its own share (plane)



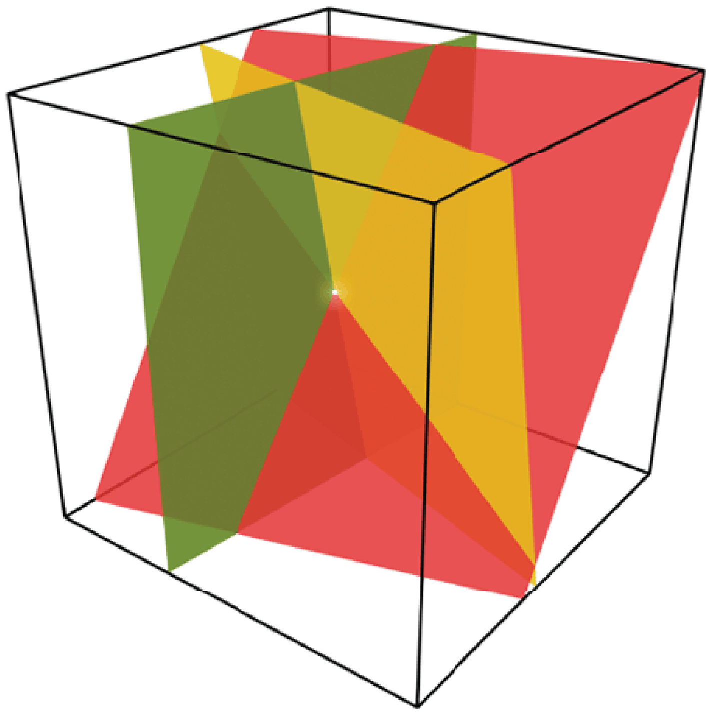
BLAKLEY SECRET SHARING

But two parties can narrow the secret down to a line!



BLAKLEY SECRET SHARING

Secret is the intersection point



VERIFIABLE SECRET SHARING

- There is reconstructable secret even if the dealer is malicious
- How can I know whether a share is correct or not?
  - Note that the correctness of a share can be verified using t other shares.
  - However, we can't ask other parties to reveal t shares.
- So each share should have a commitment which is public.
  - The correctness of shares can be verified using commitments.
  - This is called Verifiable Secret Sharing (VSS).

VERIFIABLE SECRET SHARING

EXAMPLES

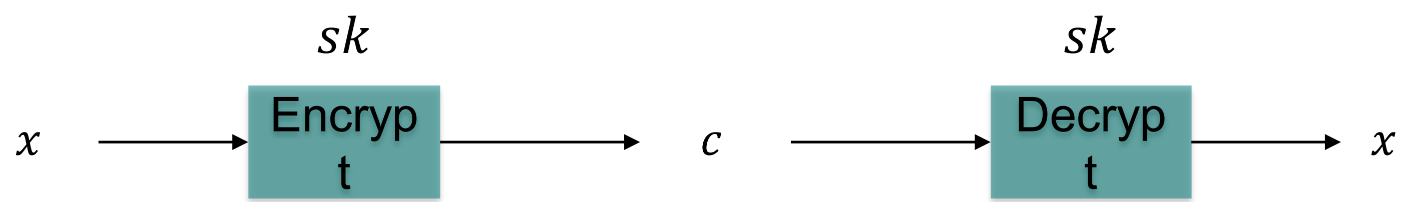
- Feldman's scheme: based on Shamir's, but commitments to the coefficients of $f(x) = s + a_1 x + a_2 x^2 + \cdots$ are also distributed: g^s, g^{a_1}, g^{a_1} ...
  - To verify that your share $(i, f(i))$, where $f(i) = y$, is really a point of the polynomial: $g^y \stackrel{?}{=} c_0^{i^0}\, c_1^{i^1} \cdots c_t^{i^t} = \prod_{j=0}^{t} c_j^{i^j} = \prod_{j=0}^{t} g^{a_j i^j} = g^{\sum_{j=0}^{t} a_j i^j} = g^{f(i)}$
- Benaloh's scheme: interactive, based on Shamir's, participants can (probabilistically) prove that all the shares are collectively t-consistent (any t shares yield the same polynomial)
- Publicly verifiable secret sharing: anybody can verify that the participants received correct shares.
  - E.g.: Chaums and Pedersen Scheme

Functional Encryption

SYMMETRIC ENCRYPTION VS PUBLIC-KEY ENCRYPTION

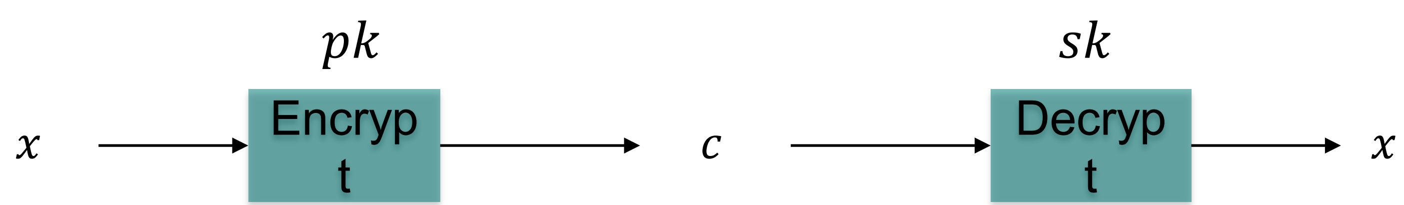
Symmetric Encryption

$sk \leftarrow KeyGen(1^{\lambda})$



Public-Key Encryption

$(sk, pk) \leftarrow KeyGen(1^{\lambda})$

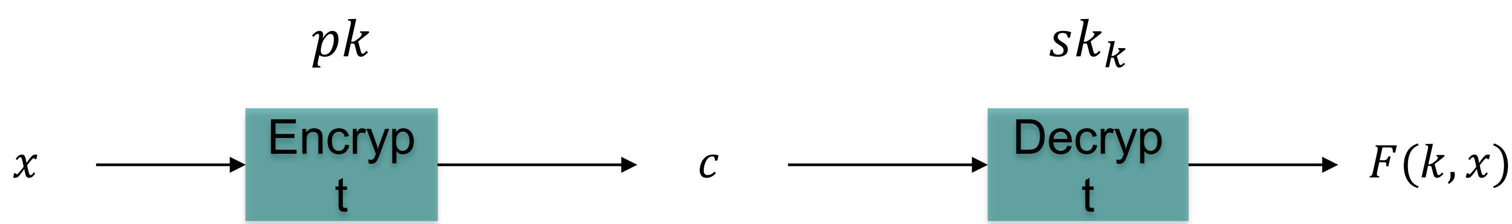


FUNCTIONAL ENCRYPTION (FE)

More Advanced Concept Of Encryption

- A FE scheme for a functionality " $F: K \times X \longrightarrow \{0,1\}^*$ " enables to evaluate $F(k, x)$ given the encryption of x
  - K: key space
  - X: plaintext space

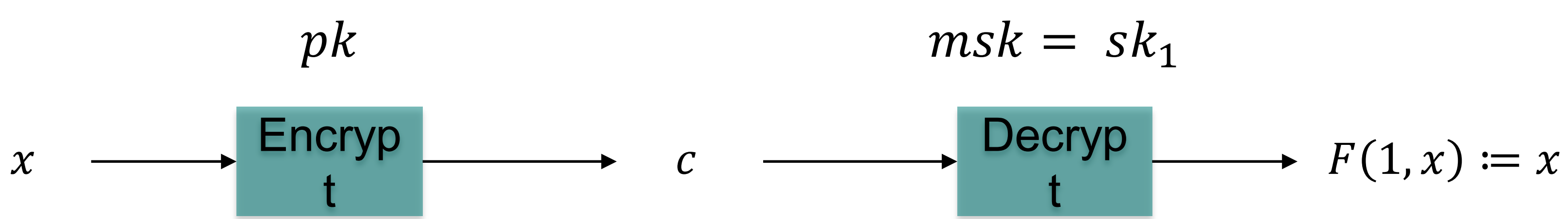
- $(msk, pk) \leftarrow Setup(1^\lambda)$ master secret key and public key
- $sk_k \leftarrow KeyGen(msk, k)$ for $k \in K$



* Boneh, Dan, Amit Sahai, and Brent Waters. "Functional encryption: Definitions and challenges." *Theory of Cryptography Conference*. Springer, Berlin, Heidelberg, 2011.

STANDARD PKE IS A SPECIAL CASE OF FE

- PKE is a FE scheme for the functionality $F(1, x) \coloneqq x$
- $(msk, pk) \leftarrow Setup(1^\lambda)$
- $sk_1 \leftarrow KeyGen(msk, 1)$



PREDICATE ENCRYPTION (PE)

Sub-Class of FE

- In many applications, $x \in X$ itself is a pair (ind, m) index and message
- PE is a sub-class of FE where
  - Plaintext space X has an additional structure $X \coloneqq I \times M$
  - PE defines an additional relation called "Predicate" $P: K \times I \longrightarrow \{1,0\}$
    - M: payload message space
    - I: index space; could be also a ciphertext attribute space
    - K: key space; could be also a key attribute space
  - The FE functionality F is defined as

$$F(k \in K, (ind, m) \in X) \coloneqq \begin{cases} m & \text{if } P(k, ind) = 1 \\ \perp & \text{if } P(k, ind) = 0 \end{cases}$$

- There are two types: PE with private index and PE with public index

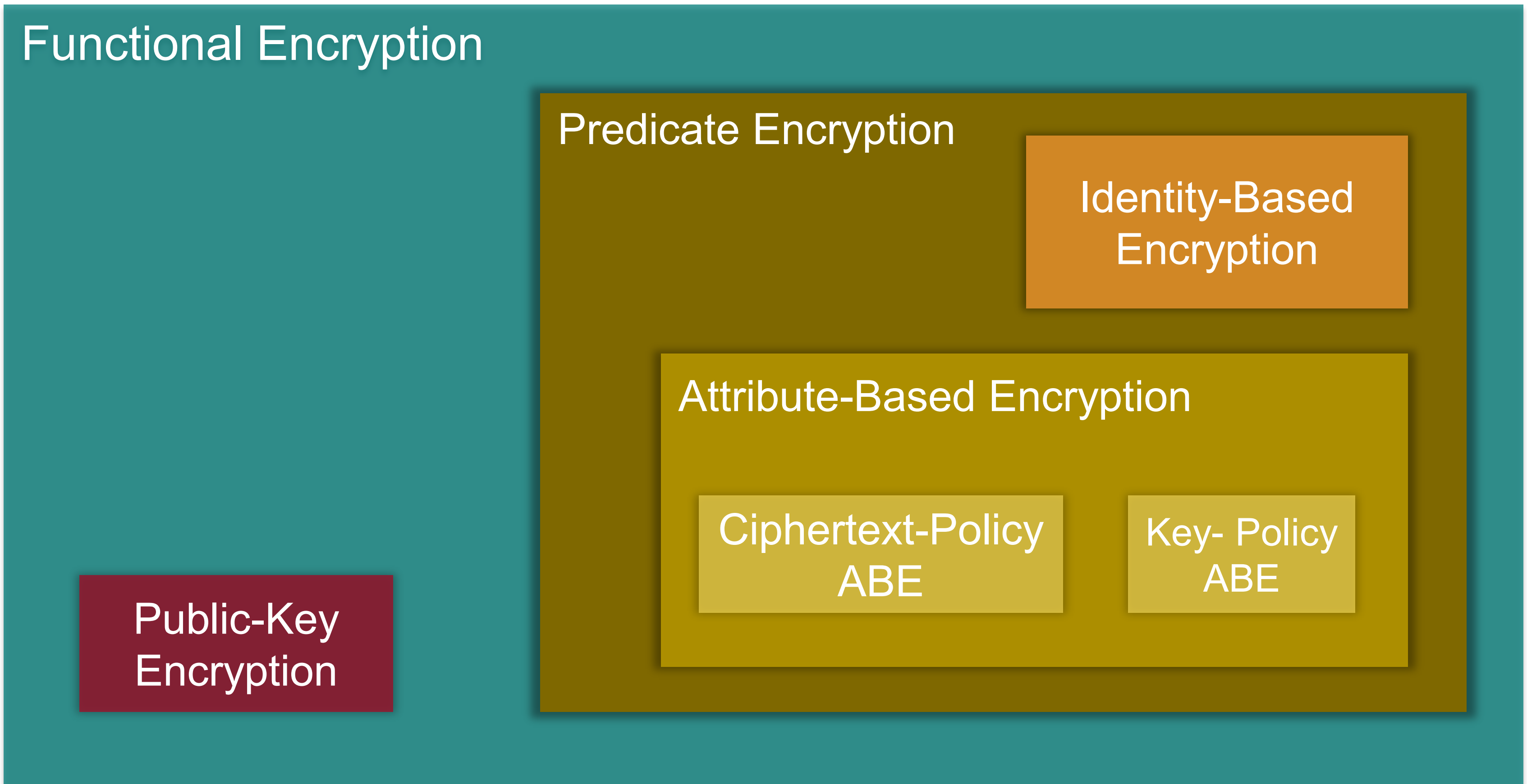
EXAMPLES OF PE SCHEMES

- PE with public index
  - IBE: **I**dentity-**B**ased **E**ncryption where “ $P \Leftrightarrow =$ ”
  - ABE: **A**ttribute-**B**ased **E**ncryption where “ $P \Leftrightarrow$ a combination of $\wedge$ and $\vee$ ”
    - Key Policy ABE
    - Ciphertext-Policy ABE.

- PE with private index
  - HVE: **H**idden **V**ector **E**ncryption where “ $P \Leftrightarrow (* \cdots *) = (* \cdots *)$ ”
  - IPPE: **I**nner **P**roduct **P**redicate **E**ncryption where “ $P \Leftrightarrow \perp$ ”

* Boneh, Dan, Amit Sahai, and Brent Waters. "Functional encryption: Definitions and challenges." *Theory of Cryptography Conference*. Springer, Berlin, Heidelberg, 2011.

Functional Encryption Overview

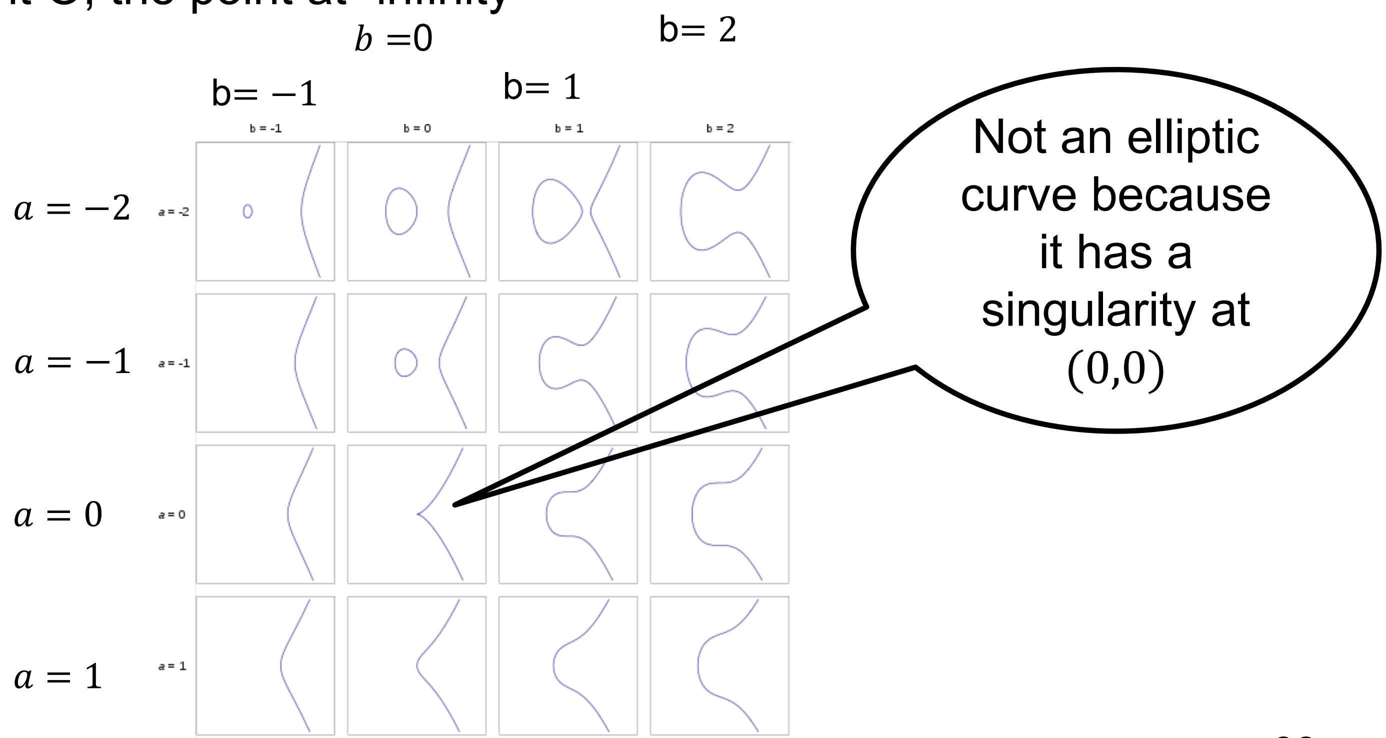


Elliptic Curves

ELLIPTIC CURVE CRYPTO (ECC)

ELLIPTIC CURVES

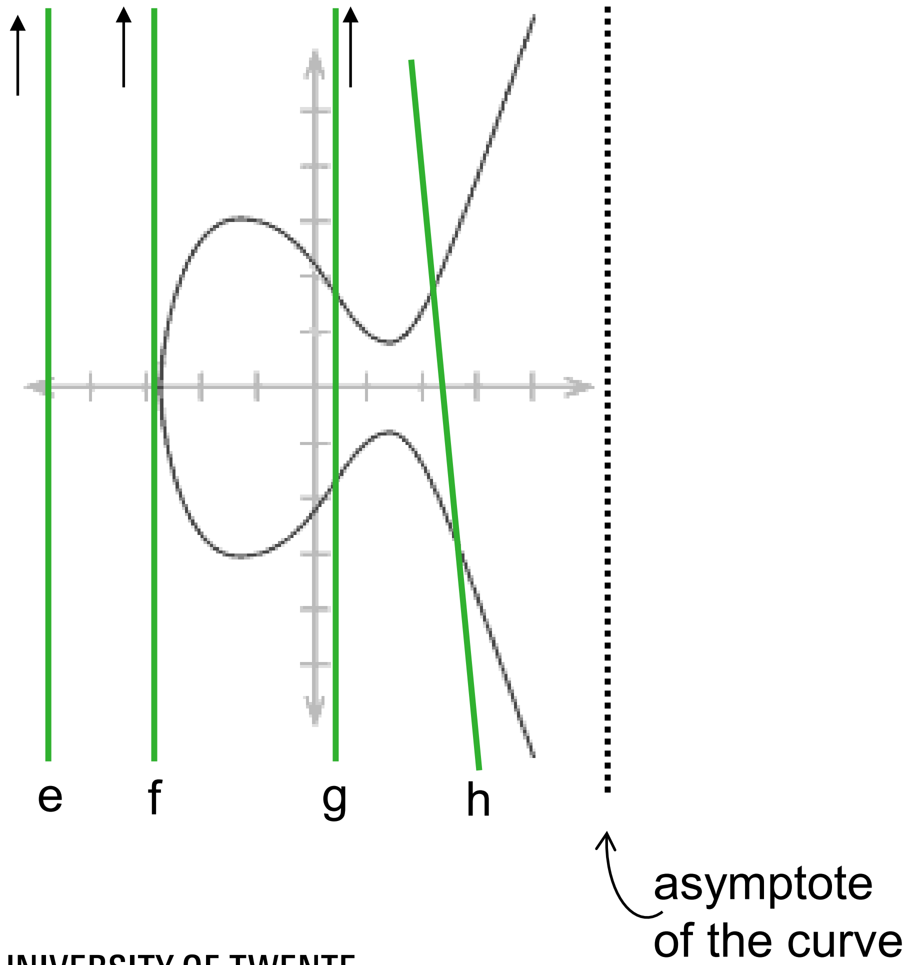
- An elliptic curve is a non-singular algebraic curve on the plane. It consists of points (x, y) for which $y^2 = x^3 + ax + b$ where $(a, b) \neq (0,0)$, and a special point O, the point at "infinity"

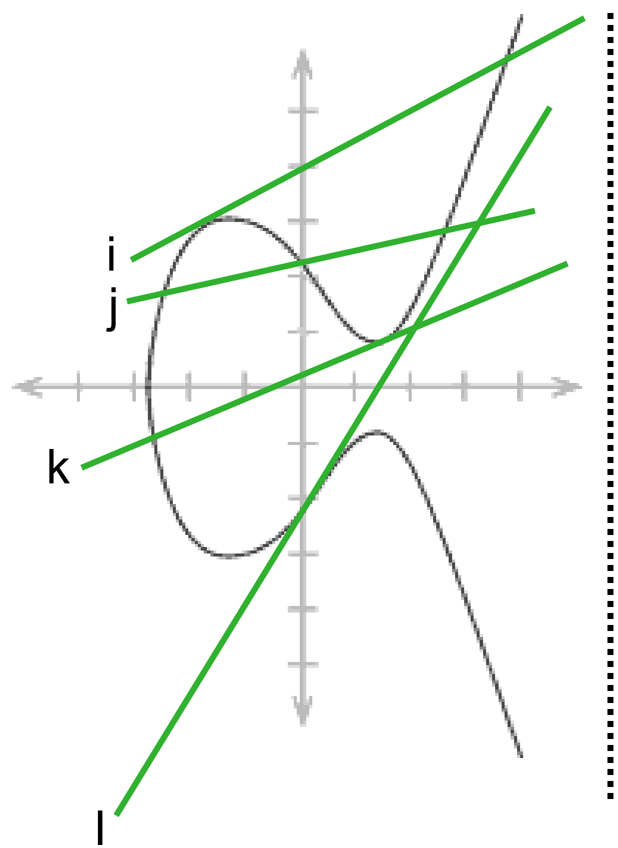


ECC: ELLIPTIC CURVE PROPERTIES

- EC consists of points (x, y) for which $y^2 = x^3 + ax + b$ where $(a, b) \neq (0,0)$, and a special point O, the point at “infinity”
  - y is present with only even exponents => the curve is symmetric to axis x
  - every line intersects the curve in 3 points. Exception: vertical lines that only intersect the curve at O, and tangents at inflection points
    - tangent points have multiplicity 2
    - see image on next slide

LINES INTERSECTING ELLIPTIC CURVES





ECC: ELLIPTIC CURVE GROUP

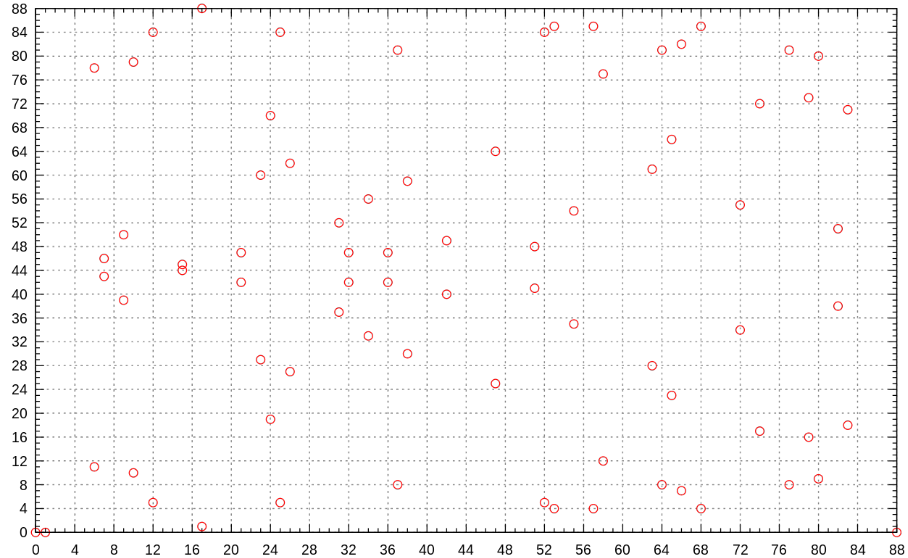
- its points form a group with a special operation (noted as +):
  - the neutral element is O (so -O=O)
  - for a point P= (p_0, p_1), its inverse -P is $(p_0, -p_1)$, its image mirrored to axis x (note that this is always on the curve)
  - for any two points P,Q on the curve, P+Q=-R, where R is the third point where the line $\overline{PQ}$ and the curve intersect. This means:
    - P+-P=O, which is what we expect anyway
    - if $\overline{PQ}$ is a tangent at Q then P+Q=-Q
  - if P=Q, define $\overline{PQ}$ as the tangent
  - if P=Q is an inclination point, then R=P, so P+Q=-P

ECC: ELLIPTIC CURVES USED FOR CRYPTOGRAPHY

- in implementations, O is not infinity but a chosen max point
- prime curve: max point is a prime
- showing the whole number points, it looks very different (due to wrapping)
- unlike in factoring for example, in ECC, there is no better “trapdoor” than the naïve method

Important property:

- kP=P+P+…+P is easy to compute, but
- given P and Q, it is very hard to find k for which Q=kP
  - unlike in case of factorization, there is no better algorithm than the naïve approach



Elliptic curve integer points

ECC: THE MAIN ADVANTAGE

- Elliptic curves are more convenient than Galois fields for public key encryption:
  - the difference in computational complexity between encryption/decryption and finding the private key is much larger for elliptic curves than for factoring
  - => in case of equally efficient schemes, the one based on elliptic curves gives much better security