MTAT.07.014 Cryptographic Protocols

Helger Lipmaa

University of Tartu

MTAT.07.014 Cryptographic Protocols

Last modified: November 8, 2011

Outline I

1. Homomorphic Protocols: Beginning
   - First Lecture: Introduction
   - Second Lecture: Elgamal
   - Third Lecture: MH Protocols. Security
   - Fourth Lecture: Additively Homomorphic Encryption
2. Semisimulatability ++
   - Fifth Lecture. Semisimulatability

References I

Aiello, W., Ishai, Y., and Reingold, O. (2001).
Priced Oblivious Transfer: How to Sell Digital Goods.
In Pfitzmann, B., editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135, Innsbruck, Austria. Springer-Verlag.

Cramer, R., Gennaro, R., and Schoenmakers, B. (1997).
A Secure and Optimally Efficient Multi-Authority Election Scheme.
In Fumy, W., editor, *EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 103–118, Konstanz, Germany. Springer-Verlag.

Damgård, I. and Jurik, M. (2001).
A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System.
In Kim, K., editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136, Cheju Island, Korea. Springer-Verlag.

Elgamal, T. (1985).
A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.
IEEE Transactions on Information Theory, 31(4):469–472.

Gentry, C. and Ramzan, Z. (2005).
Single-Database Private Information Retrieval with Constant Communication Rate.
In Caires, L., Italiano, G. F., Monteiro, L., Palamidessi, C., and Yung, M., editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 803–815, Lisboa, Portugal. Springer-Verlag.

References II

Laur, S. and Lipmaa, H. (2007).
A New Protocol for Conditional Disclosure of Secrets And Its Applications.
In Katz, J. and Yung, M., editors, *ACNS 2007*, volume 4521 of *LNCS*, pages 207–225, Zhuhai, China. Springer-Verlag.

Lipmaa, H. (2009).
First CPIR Protocol with Data-Dependent Computation.
In Lee, D. and Hong, S., editors, *ICISC 2009*, volume 5984 of *LNCS*, pages 193–210, Seoul, Korea. Springer-Verlag.

Naor, M. and Pinkas, B. (1999).
Oblivious Transfer And Polynomial Evaluation.
In *STOC 1999*, pages 245–254, Atlanta, Georgia, USA. ACM Press.

Paillier, P. (1999).
Public-Key Cryptosystems Based on Composite Degree Residuosity Classes.
In Stern, J., editor, *EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 223–238, Prague, Czech Republic. Springer-Verlag.

First Lecture: Introduction

Preliminaries

- I assume you have seen different primitives
  - Block ciphers, stream ciphers
  - Hash functions
  - Public-key cryptosystems
  - Signature schemes

  (Crypto I or an equivalent course. . .)
- For every type of primitive, you have hopefully seen some representatives, a security definition, and sometimes an attack showing that the representatives are not secure

Goal of Cryptographic Protocols

- More and more activities are done online
  - Examples: e-voting, digital signatures
- Some activities are completely new/on a completely new scale
  - Example: (privacy-preserving) data mining
- In all such cases, one should get security/correctness and privacy in the presence of malicious parties

Def. of Cryptographic Protocols

- Cryptographic protocol: a two/multi-party protocol that achieves its goals and protects privacy even in the presence of realistically malicious parties

Why It May Be Hard: CPIR I

- Server has database $\vec{f} = (f_1, \ldots, f_n)$, $|f_i| = \ell$
- Client has index $x \in \{1, \ldots, n\}$
- Computationally-Private Information Retrieval:
  - Client should obtain f_x (and may be more)
  - Server should obtain no new information
    - Nothing about x!
- Simple protocol: server sends $\vec{f}$ to client
  - Takes ℓn bits, too expensive in practice
- Can it be done better?

Why It May Be Hard: CPIR II

- If no privacy needed:
  - Client sends x, $|x| = \lceil \log_2 n \rceil$, to server
  - Server sends f_x, $|f_x| = \ell$, to client
  - $\lceil \log_2 n \rceil + \ell$ bits
  - Very small constant $\Theta(1)$ computation on modern computer
- What if privacy needed?
- Communication can be cut down to $\Theta(\log n + \ell + \kappa)$ [Gentry and Ramzan, 2005]
  - κ is security parameter (e.g., key length)
- What about computation?

Why It May Be Hard: CPIR III

- **"Theorem":** since server does not know which index client obtains, server has to "touch" all database elements. $\Theta(n)$ computation
- It was thought a few years ago that this is it
- [Lipmaa, 2009]: $\Theta(n)$ computation can be done in preprocessing phase, online computation can be decreased to $O(n/\log n)$ and often less
- Preprocessing is still $\Theta(n)$ as compared to $\Theta(1)$ in non-private case ☹

Why Often Simpler Than Assumed I

- In e-voting, server receives ciphertexts of individual ballots, and outputs a plaintext tally
- Goal: tally is correct but server does not know anything extra about individual ballots
- Sounds impossible?
- Can be done if one can do arithmetics on ciphertexts: one server "adds up" ballots and second server decrypts "'"sum"

Why Often Simpler Than Assumed II

- In e-voting, server must prove that his actions were correct, without revealing any extra information
- Sounds impossible?
- Can be done by using zero-knowledge and proven with simulation-based proofs

Simple Example: Veto

- Assume Alice and Bob have to decide on some issue
- Vetoing: decision taken only if everybody supports it
- Privacy: minimal amount of information about votes will be leaked
  - If Alice votes for then the result will be equal to Bob's vote $\Rightarrow$ Bob's privacy cannot be protected here
  - If Alice votes against then result will be "no" independently of Bob's input $\Rightarrow$ Alice should get no information

Mathematical Formulation: Veto = AND

- Assume the private inputs are $a, b \in \{0, 1\}$
- The common output is $f(a, b) := a \wedge b$
- Alice/Bob should not get to know more than inferred from her/his private input and $f(a, b)$
- In general case, every party can have a different private output $f_i(x_1, \dots, x_n)$
- Then the task is:
  - given private inputs b_i, party i should learn $f_i(b_1, \dots, b_n)$ and nothing else

Example 2: Scalar Product

- Alice's input is $\vec{a} = (a_1, \ldots, a_n)$, Bob's input is $\vec{b} = (b_1, \ldots, b_n)$
- Alice's output: $f(\vec{a}, \vec{b}) = \sum_{i=1}^{n} a_i \cdot b_i$
- Bob's output: $\perp$ (nothing)
- Alice should be convinced that her output is correct

Example 3: E-voting

- n voters v_i, m candidates c_j
- Simple case: All voters cast v_i their ballots for some candidate c_j, $b_i = c_j$
- Ballots are sent to voting servers who output the tally: for each $j \in \{1, \dots, m\}$, $T_j = |\{i \in [n] : b_i = c_j\}|$
- Everybody should learn $\{T_j : j \in \{1, \dots, m\}\}$
- Nobody should learn anything else
- Voters should be convinced the result is correct

Definitions of Security

- Will be postponed — we will first see some natural protocols
- Semihonest model: parties behave honestly, but are curious
  - Security = privacy (in semihonest model)
- Malicious model: parties behave adversarially
  - Security = privacy + correctness
  - Will study later

Efficient Protocols Based on Algebra

- Many efficient protocols are based on algebraic structures
- Common example: a finite cyclic group $(\mathbb{G}, \circ)$ where the exponentiation $\phi : \mathbb{Z}_q \to \mathbb{G}$ is both one-way (hard to invert) and an isomorphism:

$$g^0 = 1 \ , \quad g^{-a} = 1/g^a \ , \quad g^a g^b = g^{a+b} \ .$$

- One-way exponentiation makes it possible to design very efficient protocols for many problems.

Reminder: Groups

$(\mathbb{G}, \circ)$ is a group if:

- $\mathbb{G}$ is set, $\circ : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ is binary operation
- Associative: $g_1 \circ (g_2 \circ g_3) = (g_1 \circ g_2) \circ g_3$
- Exists $1 \in \mathbb{G}$, s.t. for all g, $1 \circ g = g \circ 1 = g$
- $\forall g \exists g^{-1} \in \mathbb{G}$, s.t. $g \circ g^{-1} = g^{-1} \circ g = 1$

$(\mathbb{G}, \circ)$ is abelian if additionally $g_1 \circ g_2 = g_2 \circ g_1$ for all g_1, g_2

- Multiplicative group: $\cdot$, 1, g^{-1}
- Additive group: $+$, 0, $-g$

Reminder: Cyclic groups

- Let $(\mathbb{G}, \circ)$ be a group
- $g^x = g \cdot g \cdot \dots \cdot g$ (x times)
  - If $x = \sum 2^i x_i$ then $g^x = g^{\sum 2^i x_i} = \prod (g^{2^i})^{x_i}$
- $g^{-x} = g^{-1} \cdot g^{-1} \cdot \dots \cdot g^{-1}$
- For $g \in \mathbb{G}$, let $\langle g \rangle := \{g^x : x \in \mathbb{Z}\}$
- g is a generator of $\langle g \rangle$
- If $\mathbb{G} = \langle g \rangle$ then $\mathbb{G}$ is cyclic
- **Example:**
  - $(\mathbb{Z}, +)$ is cyclic with generator 1
  - $(\mathbb{Z}_q = \{0, 1, \dots, q-1\}, +)$ is cyclic with gen. 1

Reminder: Group Order

- Element $g \in \mathbb{G}$ has order $q = \mathrm{ord}(g)$ if $g^q = 1$ and $g^i \neq 1$ for $0 < i < q$
- Group $\mathbb{G}$ has order q, $q = \mathrm{ord}(\mathbb{G})$ if $q = \max_{g \in \mathbb{G}} \mathrm{ord}(g)$
- If $\mathbb{G}$ is cyclic of order q, then for every generator $g, h \in \mathbb{G}$, there exists a unique $i \in \mathbb{Z}_q$, such that $h = g^i$
- Note that if $q = \mathrm{ord}(\mathbb{G})$, then $\forall i : g^i = g^{i \bmod q}$

Reminder: Divisibility Etc

- For $a, b \in \mathbb{Z}$, $a \mid b$ if there exists $c \in \mathbb{Z}$ such that $b = ca$
- For $a, b > 1$, $\gcd(a, b)$ is the greatest common divisor of a and b
  - $\gcd(a, b) \mid a$, $\gcd(a, b) \mid b$
  - If $c \mid a$ and $c \mid b$, then $c \leq \gcd(a, b)$
- If $\gcd(a, b) = 1$, then a and b are coprime
- $\gcd(a, b)$ can be computed efficiently by using the Euclidean Algorithm

Instantiation 1 of $\mathbb{G}$

- For $n > 1$,
  $\mathbb{Z}_n^* := \{i \in \{1, \dots, n-1\} : \gcd(n, i) = 1\}$
- Fact: i is reversible in $(\mathbb{Z}_n, \cdot)$ iff $\gcd(n, i) = 1$
  - $(\mathbb{Z}_n^*, \cdot)$ is group
- $\varphi(n) := |\mathbb{Z}_n^*|$ is Euler's totient function
- If p is prime, then $\varphi(p) = p - 1$
  - $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$
- Lagrange's theorem: If $\mathbb{G}$ is finite and $\mathbb{G}' \subseteq \mathbb{G}$ is subgroup, then $\mathrm{ord}(\mathbb{G}') \mid \mathrm{ord}(\mathbb{G})$
- OTOH: If $q \mid p$ and $\mathbb{G}$ is group of order p, then $\mathbb{G}$ has subgroup of order q

Instantiation 1 of $\mathbb{G}$

Example

Let p, q be two large primes s.t. $q \mid (p-1)$. Let $\mathbb{G}$ be the unique subgroup of $\mathbb{Z}_{p^*}$ of order q. Let g be the generator of $\mathbb{G}$.

Explanation: $|\mathbb{Z}_p^*| = p - 1$, thus there exists (unique) subgroup $\mathbb{G}$ of $\mathbb{Z}_p^*$ of order q.
In practical instantiations, $\log_2 p \approx 1536$ and $\log_2 q \approx 160$. We need 1536 bits to represent an element of $\mathbb{G}$. Exponentiation in $\mathbb{G}$ takes up to 160 multiplications.

Instantiation 2 of $\mathbb{G}$

The most popular alternative involves elliptic curve groups, where $\log_2 q = 160$ and $\mathbb{G}$ can be represented by using $\approx \log_2 q$ bits. Much more efficient than the previous case, though also much more complicated mathematics.

Fineprint: The elliptic curve groups must be chosen carefully. For example, in some e.c. groups, one can efficiently solve DDH problem. But such groups are useful otherwise.

Abstracting $\mathbb{G}$

In the next, we will abstract away the concrete group and assume that $\mathbb{G}$ is a multiplicative cyclic group of order q (with some hardness assumptions).

Second Lecture: Elgamal

See [Elgamal, 1985] for original paper on Elgamal cryptosystem.

Reminder: group isomorphisms

- Let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \cdot)$ be groups
- Function $f : \mathbb{G}_1 \to \mathbb{G}_2$ is group isomorphism, if
  - $f(g_1 + g_2) = f(g_1) \cdot f(g_2)$
  - $f(0) = 1$
  - $f(-g) = f(g)^{-1}$

Discrete Logarithm Problem

- Let $\mathbb{G}$ be cyclic group of prime order q
- Efficiently computable isomorphism $f(a) : \mathbb{Z}_q \to \mathbb{G}$: given a generator g, $a \mapsto g^a =: f(a)$.
  - f is an isomorphism: $f(a) \cdot f(b) = g^a g^b = g^{a+b} = f(a+b)$, $f(0) = g^0 = 1$, $f(-a) = g^{-a} = 1/g^a = f(a)^{-1}$
- Discrete Logarithm Assumption: f^{-1} is intractable to compute. I.e., given (g, g^a), it is difficult to find a.

Reminder: Basic Complexity Theory

- Parameter: input size κ
- $poly(\kappa) = \kappa^{O(1)}$: polynomial in κ, exists polynomial f such that $|poly(\kappa)| \leq |f(\kappa)|$
- $negl(\kappa) = \kappa^{-\omega(1)}$: negligible in κ, for every polynomial f, $|poly(\kappa)| < |f^{-1}(\kappa)|$
- "Efficient" algorithm: works in time $poly(\kappa)$
- Probabilistic algorithm can use a random string
- Non-uniform algorithm: construction of algorithm for concrete input size can be inefficient

DL Assumption, More Formally

Let $\mathbb{G}$ be a cyclic group of prime order q. Fix generator $g \in \mathbb{G}$. Let

$$Adv^{dl}_{\mathbb{G}}(\mathcal{A}) := \Pr[a \leftarrow \mathbb{Z}_q : \mathcal{A}(g, g^a) = a] \ .$$

We say that $\mathbb{G}$ is (τ, ε)-DL group if for any non-uniform probabilistic adversary $\mathcal{A}$ that works in time $\leq \tau$, $Adv^{dl}_{\mathbb{G}}(\mathcal{A}) \leq \varepsilon$.
We say $\mathbb{G}$ is DL group if it is $(poly(\kappa), negl(\kappa))$-DL group.

Assumption:

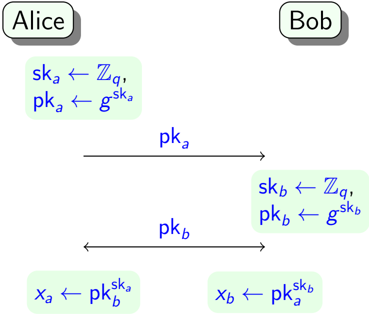
- Sampleability: it is easy to pick a random element from $\mathbb{G}$
- Follows from isomorphism: sample $a \leftarrow \mathbb{Z}_q$ (easy) and compute $b \leftarrow g^a$; since a is a random element of $\mathbb{Z}_q$, then b is a random element of $\mathbb{G}$

Diffie-Hellman Key Exchange Protocol I

- Alice and Bob have both secret keys sk_a and sk_b and public keys pk_a and pk_b
- Only Alice knows sk_a, while everybody knows pk_a. Same for Bob
- Alice and Bob generate a new common secret key x such that only Alice and Bob know it
- x is later used to encrypt other messages
- We assume that all messages are sent on authenticated channels
  - Alice's/Bob's messages are known to come from Alice/Bob

Diffie-Hellman Key Exchange Protocol II

- Fix prime q,
  s.t. $\log_2 q \approx 2 \cdot \kappa$, and
  cyclic group $\mathbb{G}$ of order q.
  Let g be generator of $\mathbb{G}$
- Protocol is on the right
- $x_a = (g^{\mathsf{sk}_b})^{\mathsf{sk}_a} = g^{\mathsf{sk}_a \cdot \mathsf{sk}_b} = (g^{\mathsf{sk}_a})^{\mathsf{sk}_b} = x_b$ and Alice and Bob have established a secret key



Security of DH Key Exchange

- Goal of adversary: given $(g, g^{\mathsf{sk}_a}, g^{\mathsf{sk}_b})$ for random $\mathsf{sk}_a, \mathsf{sk}_b \leftarrow \mathbb{Z}_q$, output $x = g^{\mathsf{sk}_a \cdot \mathsf{sk}_b}$
- This is not known to be hard under DL assumption, and thus there is separate assumption (CDH) for this problem
  - Computational Diffie-Hellman
- If CDH is hard, then clearly DL is hard
- There are some contrived groups where DL is hard but CDH is not

CDH Assumption, Formally

Let $\mathbb{G}$ be a cyclic group of prime order q. Fix generator $g \in \mathbb{Z}_q^*$. Let

$$Adv_{\mathbb{G}}^{cdh}(\mathcal{A}) := \Pr[a, b \leftarrow \mathbb{Z}_q : \mathcal{A}(g, g^a, g^b) = g^{ab}] \ .$$

We say that $\mathbb{G}$ is (τ, ε)-CDH group if for any non-uniform probabilistic adversary $\mathcal{A}$ that works in time $\leq \tau$, $Adv_{\mathbb{G}}^{cdh}(\mathcal{A}) \leq \varepsilon$.
We say $\mathbb{G}$ is CDH group if it is $(poly(\kappa), negl(\kappa))$-CDH group.

Security of DH Key Exchange, II

- Goal of adversary: given $(g, g^{\mathsf{sk}_a}, g^{\mathsf{sk}_b})$ for random $\mathsf{sk}_a, \mathsf{sk}_b \leftarrow \mathbb{Z}_q$, output $x \leftarrow g^{\mathsf{sk}_a \cdot \mathsf{sk}_b}$
- Not sufficient!
- Adversary should not get to know anything about x, i.e., x should look to her completely random
- Not known to be hard under CDH assumption, and thus there is separate assumption for this problem
  - Decisional Diffie-Hellman
  - There are well-known CDH groups that are not DDH groups

DDH Assumption, Formally

Let $\mathbb{G}$ be cyclic, prime order q. Fix gen. $g \in \mathbb{Z}_q^*$.

Experiment 1

Set $(a, b) \leftarrow \mathbb{Z}_q \times \mathbb{Z}_q$.
Set $\vec{g} \leftarrow (g, g^a, g^b, g^{ab})$.

Experiment 2

Set $(a, b, c) \leftarrow \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q$.
Set $\vec{g} \leftarrow (g, g^a, g^b, g^c)$.

$$Adv_{\mathbb{G}}^{ddh}(\mathcal{A}) := |\Pr[Exp1 : \mathcal{A}(\vec{g}) = 1] - \Pr[Exp2 : \mathcal{A}(\vec{g}) = 1]| \ .$$

$\mathbb{G}$ is (τ, ε)-DDH group if for any non-uniform probabilistic adversary $\mathcal{A}$ that works in time $\leq \tau$, $Adv_{\mathbb{G}}^{ddh}(\mathcal{A}) \leq \varepsilon$.
$\mathbb{G}$ is DDH group $\Leftrightarrow$ $(poly(\kappa), negl(\kappa))$-DDH group.

Public-Key Encryption

Public-key cryptosystem is triple of efficient algorithms $\Pi = (G, E, D)$, such that

- κ is security parameter (e.g., key length)
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$ is key generation algorithm
- $E_{\mathsf{pk}}(m; r) = c$ is randomized encryption algorithm
- $D_{\mathsf{sk}}(c) = m$ is decryption algorithm

and

Correctness: $D_{\mathsf{sk}}(E_{\mathsf{pk}}(m; r)) = m$ for all m, r and $(\mathsf{sk}, \mathsf{pk}) \in G(1^\kappa)$

Homomorphic Encryption

A public-key cryptosystem is multiplicatively homomorphic if:

- The plaintext set $(\mathcal{M}, \cdot)$ is multiplicative group, the randomizer set $(\mathcal{R}, \circ)$ is group, and the ciphertext set $(\mathcal{C}, \cdot)$ is multiplicative group.
  - All three sets can depend on $(\mathsf{sk}, \mathsf{pk})$.
- $E_{\mathsf{pk}}(m_1; r_1) \cdot E_{\mathsf{pk}}(m_2; r_2) = E_{\mathsf{pk}}(m_1 \cdot m_2; r_1 \circ r_2)$
- Thus $D_{\mathsf{sk}}(E_{\mathsf{pk}}(m_1; r_1) \cdot E_{\mathsf{pk}}(m_2; r_2)) = m_1 \cdot m_2$ for every m_1, m_2, r_1, r_2.
- Discrete logarithm problem is hard in group $\mathcal{M}$

Hom. Encryption: Basic Properties

- $D_{sk}(E_{pk}(m_1; r_1) \cdot E_{pk}(m_2; r_2)) = m_1 \cdot m_2$
  - Computation of encryption of $m_1 \cdot m_2$ does not need knowledge of m_1 or m_2
- For $m \in \mathcal{M}$ and $\alpha \in \mathbb{Z}_{|\mathcal{M}|}$, $D_{sk}(E_{pk}(m; r)^\alpha) = m^\alpha$ (by def. of exp.)
- Given x and $\{E_{pk}(g^{f_i})\}$ for $i \in \{0, \ldots, t\}$, one can compute

$$E_{pk}(g^{f(x)}) = \prod_{i=0}^{t} E_{pk}(g^{f_i})^{x^i} \quad .$$

where $f(X) := \sum_{i=0}^{t} f_i X^i$

Elgamal Encryption

Assume a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q.

- $G(1^\kappa)$: let $\mathsf{sk} \leftarrow \mathbb{Z}_q$ and $\mathsf{pk} \leftarrow h = g^{\mathsf{sk}}$.
- Encryption of $m \in \mathbb{G}$: generate random $r \leftarrow \mathbb{Z}_q$. Compute $E_{\mathsf{pk}}(m; r) \leftarrow (mh^r, g^r)$
- Decryption of $c = (c_1, c_2) \in \mathbb{G}^2$: set $D_{\mathsf{sk}}(c_1, c_2) \leftarrow c_1 / c_2^{\mathsf{sk}}$.

Correctness:

$$\begin{aligned} D_{\mathsf{sk}}(E_{\mathsf{pk}}(m; r)) =& D_{\mathsf{sk}}(mh^r, g^r) = m \cdot h^r / (g^r)^{\mathsf{sk}} \\ =& m \cdot (g^{\mathsf{sk}})^r / (g^{\mathsf{sk}})^r = m \quad . \end{aligned}$$

Elgamal Encryption is Homomorphic

Homomorphism in cyclic group $\mathbb{G}$ of order q, where DL is assumed to be hard. Ciphertext group is $\mathbb{G}^2$ with $(g_1, g_1') \cdot (g_2, g_2') = (g_1 g_2, g_1' g_2')$

$$\begin{aligned} E_{\mathrm{pk}}(m_1; r_1) \cdot E_{\mathrm{pk}}(m_2; r_2) =& (m_1 m_2 h^{r_1+r_2}, g^{r_1+r_2}) \\ =& E_{\mathrm{pk}}(m_1 \cdot m_2; r_1 + r_2) \enspace . \end{aligned}$$

Also, for known α,

$$E_{\mathrm{pk}}(m; r)^{\alpha} = (m^{\alpha} h^{\alpha r}, g^{\alpha r}) = E_{\mathrm{pk}}(m^{\alpha}; \alpha r) \enspace .$$

Example Protocol: Asymmetric Veto

- Alice learns if $a \wedge b = 1$, Bob learns nothing
- Comp. DL is easy
- In semihonest model, Alice learns nothing except $a \wedge b$, if Elgamal is secure

Alice (a)		Bob (b)
$(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$, $r \leftarrow \mathcal{R}$		
	$\xrightarrow{(\mathsf{pk}, E_{\mathsf{pk}}(g^a; r))}$	
		$c \leftarrow E_{\mathsf{pk}}(g^a; r)^b = E_{\mathsf{pk}}(g^{ab}; br)$
	$\xleftarrow{c}$	
$m \leftarrow DL(D_{sk}(c)) = DL(g^{ab}) = ab$		

IND-CPA Security

Assume $\Pi = (G, E, D)$. Let $\mathcal{A}$ be efficient adversary.

Experiment 1

Set $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$.
Obtain $(m_1, m_2) \leftarrow \mathcal{A}(\mathsf{pk})$.
Output $E_{\mathsf{pk}}(m_1; r)$ for $r \leftarrow \mathcal{R}$.

Experiment 2

Set $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$.
Obtain $(m_1, m_2) \leftarrow \mathcal{A}(\mathsf{pk})$.
Output $E_{\mathsf{pk}}(m_2; r)$ for $r \leftarrow \mathcal{R}$.

$$Adv_{\Pi}^{cpa}(\mathcal{A}) := \left| \Pr[\mathrm{Exp1} : \mathcal{A} = 1] - \Pr[\mathrm{Exp2} : \mathcal{A} = 1] \right| .$$

Π is IND-CPA secure if no efficient $\mathcal{A}$ has non-negligible $Adv_{\Pi}^{cpa}(\mathcal{A})$.

Elgamal Is IND-CPA Secure

Theorem

Assume that $\mathbb{G}$ is DDH-group. Then Elgamal is IND-CPA secure.

For proof, we note that if $(g_1, g_2, g_3, g_4) = (g, g^a, g^b, g^{ab})$ then $(g_4, g_3) = (g^{ab}, g^b)$ is encryption of 1 under public key $\mathsf{pk} = g_2 = g^a$.

OTOH, if $(g_1, g_2, g_3, g_4) = (g, g^a, g^b, g^c)$ for random c, then $(g_4, g_3) = (g^c, g^b) = (g^{c-ab} g^{ab}, g^b)$ is encryption of random plaintext g^{c-ab} under public key $\mathsf{pk} = g_2 = g^a$.

Elgamal Is IND-CPA Secure: Proof I I

Assume that $\mathcal{A}$ can break IND-CPA security with probability ε. Construct the next DDH distinguisher $\mathcal{D}$. (This shows that if DDH is hard, then Elgamal is IND-CPA secure.)

Elgamal Is IND-CPA Secure: Proof II I

Main idea of the proof: $\mathcal{D}$ participates in DDH "game" with challenger. Since $\mathcal{A}$ can break IND-CPA of Elgamal, $\mathcal{D}$ can use "help" from $\mathcal{A}$. Help consists in interacting with $\mathcal{A}$ in conversation that looks like IND-CPA game to $\mathcal{A}$. Thus, $\mathcal{A}$ will "break" IND-CPA of Elgamal inside that game with probability ε.

Elgamal Is IND-CPA Secure: Proof II II

Challenger | $\mathcal{D}$ | $\mathcal{A}$

$b_{ddh} \leftarrow \{1, 2\}$,
$g_1 \leftarrow \mathbb{G}, (a, b, c) \leftarrow \mathbb{Z}_q^3$,
$g_2 \leftarrow g_1^a, g_3 \leftarrow g_1^b$,
$g_4 \leftarrow (b_{ddh} = 1) \, ? \, g_1^{ab} : g_1^c$

Challenger $\xrightarrow{(g_1, g_2, g_3, g_4)}$ $\mathcal{D}$

$\mathcal{D} \xrightarrow{\text{Message } 1} \mathcal{A}$

$\mathcal{D} \xleftrightarrow{\text{Messages}} \mathcal{A}$

$\mathcal{D} \xleftarrow{\text{Message } s} \mathcal{A}$

Challenger $\xleftarrow{b'_{ddh}}$ $\mathcal{D}$

$b'_{ddh} \stackrel{?}{=} b_{ddh}$

Elgamal Is IND-CPA Secure: Proof IV

$\mathcal{D}(g_1, g_2, g_3, g_4)$ | $\mathcal{A}$

$g \leftarrow g_1$, $\mathsf{pk} \leftarrow g_2$ $\longrightarrow$

$(m_1, m_2) \leftarrow \mathcal{A}(g, \mathsf{pk})$

$\longleftarrow$ (m_1, m_2)

$b_{cpa} \leftarrow \{1, 2\}$,
$(c_1, c_2) \leftarrow (m_{b_{cpa}} \cdot g_4, g_3)$

(c_1, c_2) $\longrightarrow$

$b'_{cpa \leftarrow \mathcal{A}(g, \mathsf{pk}, (c_1, c_2))}$

$\longleftarrow$ b'_{cpa}

$b'_{ddh = (b'_{cpa} = b_{cpa})\,?\,1:2}$

Elgamal is IND-CPA Secure: Proof V

$$
\begin{aligned}
\Pr[\mathcal{D} \text{ is correct}] &= \Pr[b'_{ddh} = b_{ddh}] \\
=&\Pr[b'_{ddh} = 1 : b_{ddh} = 1] \Pr[b_{ddh} = 1] + \\
&\Pr[b'_{ddh} = 2 : b_{ddh} = 2] \Pr[b_{ddh} = 2] \\
=&\frac{1}{2} \cdot \Pr[b'_{cpa} = b_{cpa} : b_{ddh} = 1] + \frac{1}{2} \cdot \Pr[b'_{cpa} \neq b_{cpa} : b_{ddh} = 2] \\
=&\frac{1}{2} \cdot \varepsilon + \frac{1}{2} \cdot \frac{1}{2} = \frac{\varepsilon}{2} + \frac{1}{4} \enspace .
\end{aligned}
$$

Thus if $\mathcal{A}$ is successful, then $\mathcal{D}$ is successful with approximately same time and success probability.
QED

Third Lecture: MH Protocols. Security

Homomorphic Encryption: Blinding

- Let $E_{\mathrm{pk}}(m; \mathcal{R})$ be distribution that one gets by first choosing $r \leftarrow \mathcal{R}$ and then outputting $E_{\mathrm{pk}}(m; r)$
- Rerandomization/blinding: For any $m \in \mathcal{M}$ and $r \in \mathcal{R}$,

$$E_{\mathrm{pk}}(m; r) \cdot E_{\mathrm{pk}}(1; \mathcal{R}) = E_{\mathrm{pk}}(m; \mathcal{R}) \enspace .$$

- Holds since $\mathcal{R}$ is cyclic, sampleable group
- Used in situations where revealing r might compromise privacy

Example Protocol: Scalar Product I

- Alice has $(a_1, \ldots, a_t) \in \mathbb{Z}_q^t$
- Bob has $(b_1, \ldots, b_t) \in \mathbb{Z}_q^t$
- Alice learns $\sum_{i=1}^{t} a_i b_i \mod q \in \mathbb{Z}_q$
- Privacy in semihonest model:
  - Alice learns nothing else, Bob learns nothing

Example Protocol: Scalar Product II

- Comp. DL is easy if a_i, b_i are Boolean (Alice's output is $\leq t$)
- r is used for blinding: c is a random encryption of g^m

Alice $(a_1, \ldots, a_t)$ | Bob $(b_1, \ldots, b_t)$

Alice: $(\mathrm{sk}, \mathrm{pk}) \leftarrow G(1^\kappa)$, $(r_1, \ldots, r_t) \leftarrow \mathcal{R}^t$, $c_i \leftarrow E_{pk}(g^{a_i}; r_i)$

$$\xrightarrow{(\mathrm{pk}, (c_1, \ldots, c_t))}$$

Bob: $r \leftarrow \mathcal{R}$, $c \leftarrow \prod_{i=1}^t c_i^{b_i} \cdot E_{\mathrm{pk}}(1; r)$

$$\xleftarrow{c}$$

Alice: $m \leftarrow \log_g(D_{sk}(c))$

Correctness: Scalar Product Protocol

Recall $c_i = E_{\mathrm{pk}}(g^{a_i}; r_i)$. Clearly,

$$c = \prod_{i=1}^{t} c_i^{b_i} \cdot E_{\mathrm{pk}}(1; r) = \prod_{i=1}^{t} E_{\mathrm{pk}}(g^{a_i}; r_i)^{b_i} \cdot E_{\mathrm{pk}}(1; r)$$
$$= E_{\mathrm{pk}}\left(g^{\sum_{i=1}^{t} a_i b_i}; \sum_{i=1}^{t} b_i r_i + r\right) .$$

and thus

$m = \log_g(D_{sk}(c)) = \log_g(g^{\sum_{i=1}^{t} a_i b_i}) = \sum_{i=1}^{t} a_i b_i$

Example Protocol: Hamming Distance I

- Alice has $\vec{a} := (a_1, \ldots, a_t) \in \mathbb{Z}_2^t$
- Bob has $\vec{b} := (b_1, \ldots, b_t) \in \mathbb{Z}_2^t$
- Define $w_h(\vec{a}, \vec{b}) := |\{i \in \{1, \ldots, t\} : a_i \neq b_i\}|$
- Alice learns $w_h(\vec{a}, \vec{b})$
- Privacy in semihonest model:
  - Alice learns nothing else, Bob learns nothing
- Clearly $w_h(\vec{a}, \vec{b}) := \sum_{i=1}^{t}(a_i \oplus b_i) = \sum_{i=1}^{t}(b_i + (-1)^{b_i} a_i)$:
  - $0 + (-1)^0 a_i = a_i = a_i \oplus 0$
  - $1 + (-1)^1 a_i = 1 - a_i = a_i \oplus 1$

Example Protocol: Hamming Distance II

Alice $(a_1, \dots, a_t)$

Bob $(b_1, \dots, b_t)$

$(\mathrm{sk}, \mathrm{pk}) \leftarrow G(1^\kappa)$,
$(r_1, \dots, r_t) \leftarrow \mathcal{R}^t$,
$c_i \leftarrow E_{pk}(g^{a_i}; r_i)$

$$\xrightarrow{(\mathrm{pk}, (c_1, \dots, c_t))}$$

$r \leftarrow \mathcal{R}$,
$c \leftarrow \prod_{i=1}^{t}(E_{\mathrm{pk}}(g^{b_i}; 0) \cdot c_i^{(-1)^{b_i}}) \cdot E_{\mathrm{pk}}(1; r)$

$$\xleftarrow{c}$$

$m \leftarrow \log_g(D_{sk}(c))$

Correctness: Hamming Distance Protocol

Recall $c_i = E_{\mathrm{pk}}(g^{a_i}; r_i)$. Clearly,

$$
\begin{aligned}
c =& \prod_{i=1}^{t} (E_{pk}(g^{b_i}; 0) \cdot c_i^{(-1)^{b_i}}) \cdot E_{\mathrm{pk}}(1; r) \\
=& E_{\mathrm{pk}} \left(g^{\sum_{i=1}^{t} (b_i + (-1)^{b_i} a_i)}; \sum_{i=1}^{t} (-1)^{b_i} r_i + r \right) = E_{\mathrm{pk}}(g^{w_h(\vec{a}, \vec{b})}; \dots) \enspace .
\end{aligned}
$$

and thus $m = \log_g(D_{sk}(c)) = \log_g(g^{w_h(\vec{a}, \vec{b})}) = w_h(\vec{a}, \vec{b})$

2-Message Protocols I

- 2-pessage protocol is IND-CPA secure if Bob cannot distinguish between Alice's message, corresponding to Alice's input a_1, from Alice's message, corresponding to a_2
- Similar definition to IND-CPA of PKC

Alice (a) Bob (b)

$(\mathfrak{q}, \mathit{state}) \leftarrow \mathit{Query}(1^\kappa, a)$

$\xrightarrow{\mathfrak{q}}$

$\mathfrak{r} \leftarrow \mathit{Reply}(1^\kappa, b, q)$

$\xleftarrow{\mathfrak{r}}$

$\mathfrak{a} = \mathit{Answer}(1^\kappa, a, \mathit{state}, \mathfrak{r})$

IND-CPA Security of 2-Message Protocols

Assume $\Gamma = (Query, Reply, Answer)$. Let $\mathcal{A}$ be efficient adversary.

Experiment 1

Obtain $(a_1, a_2) \leftarrow \mathcal{A}(1^\kappa)$.
Output $\mathfrak{q}$ where
$(\mathfrak{q}, state) \leftarrow Query(a_1)$.

Experiment 2

Obtain $(a_1, a_2) \leftarrow \mathcal{A}(1^\kappa)$.
Output $\mathfrak{q}$ where
$(\mathfrak{q}, state) \leftarrow Query(a_2)$.

$$Adv_\Gamma^{cpa}(\mathcal{A}) := \left| \Pr[\text{Exp1} : \mathcal{A} = 1] - \Pr[\text{Exp2} : \mathcal{A} = 1] \right| .$$

Γ is IND-CPA secure if no efficient $\mathcal{A}$ has non-negligible $Adv_\Gamma^{cpa}(\mathcal{A})$.

2-Message Homomorphic Protocols

- a — anything (e.g., a real value)
- $m_i \in \mathcal{M}$ are functions of a
- $m_i = m_i(a)$

Alice (a) — **Bob** (b)

Alice: $(\mathrm{sk}, \mathrm{pk}) \leftarrow G(1^\kappa)$, For $i \in \{1, \dots, t\}$, $c_i \leftarrow E_{\mathrm{pk}}(m_i, r_i)$

$$\xrightarrow{(\mathrm{pk}; c_1, \dots, c_t)}$$

Bob: $\mathfrak{r} \leftarrow \mathit{Reply}(1^\kappa, b, \mathrm{pk}, c_1, \dots, c_t)$

$$\xleftarrow{\mathfrak{r}}$$

Alice: $\mathfrak{a} = \mathit{Answer}(1^\kappa, a, \mathrm{sk}, \mathrm{pk}, \mathfrak{r})$

Metatheorem: 2MHP are IND-CPA Secure

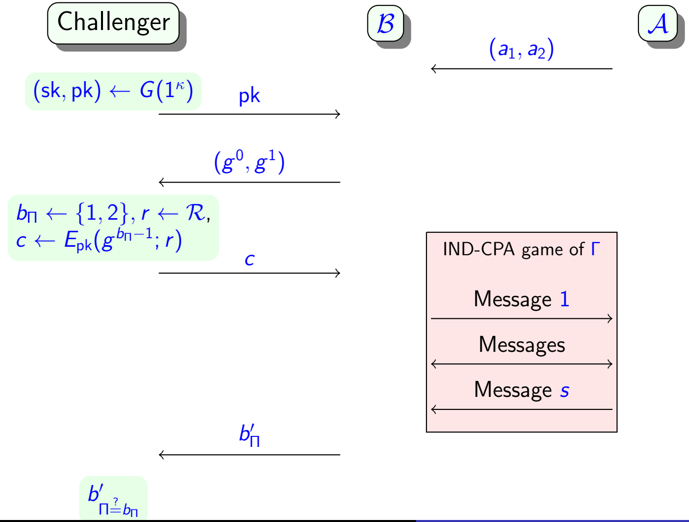
Theorem

Assume $\Pi = (G, E, D)$ *is IND-CPA secure. Then* $\Gamma = (Query, Reply, Answer)$ *is IND-CPA secure.*

Proof: 2MHP are IND-CPA Secure I

Assume $\mathcal{A}$ can break Γ with time τ and probability ε. Construct adversary $\mathcal{B}$ that breaks Π with same probability and time $\tau + 2t\tau_{\mathsf{exp}} + \mathit{small}$ as follows. (τ_{exp} is time for one exp.)

Proof: 2MHP are IND-CPA Secure II



Proof: 2MHP are IND-CPA Secure III

- $\mathcal{A}$ first gives (a_1, a_2) to $\mathcal{B}$
- Assume that if $\mathcal{B}$'s input to Γ is a_{b_Π}, then the values encrypted in Γ are $(f_1(a_{b_\Pi}), \dots, f_t(a_{b_\Pi}))$
  - In Hamming distance protocol, $f_i(\vec{a}) = a_i$
- Bob does not know $b_\Pi \in \{1, 2\}$ but he knows $E_{\mathrm{pk}}(g^{b_\Pi}; r)$ and $(f_j(a_1), f_j(a_2))$
- Clearly, $f_j(a_{b_\Pi}) = (2 - b_\Pi) f_j(a_1) + (b_\Pi - 1) f_j(a_2)$
  - $b_\Pi = 1 : (2 - 1) f_j(a_1) + (1 - 1) f_j(a_2) = f_j(a_1)$
  - $b_\Pi = 2 : (2 - 2) f_j(a_1) + (2 - 1) f_j(a_2) = f_j(a_2)$

Proof: 2MHP are IND-CPA Secure IV

- $f_j(a_{b_\Pi}) = (2 - b_\Pi) f_j(a_1) + (b_\Pi - 1) f_j(a_2)$
- $c = E_{\mathrm{pk}}(g^{b_\Pi}; r)$
- Thus $(E_{\mathrm{pk}}(g^2; 0)/c)^{f_j(a_1)} \cdot (c/E_{\mathrm{pk}}(g; 0))^{f_j(a_2)} =$
$$\underbrace{\underbrace{(\underbrace{(E_{pk}(g^2;0)/E_{\mathrm{pk}}(g^{b_\Pi};r))}_{E_{\mathrm{pk}}(g^{2-b_\Pi};-r)})^{f_j(a_1)}}_{E_{\mathrm{pk}}(g^{(2-b_\Pi)f_j(a_1)};-rf_j(a_1))} \cdot \underbrace{(E_{\mathrm{pk}}(g^{b_\Pi};r)/E_{\mathrm{pk}}(g;0))^{f_j(a_2)}}_{E_{\mathrm{pk}}(g^{(b_\Pi-1)f_j(a_2)};rf_j(a_2))}}_{E_{\mathrm{pk}}(g^{(2-b_\Pi)f_j(a_1)+(b_\Pi-1)f_j(a_2)};r(f_j(a_2)-f_j(a_1)))=E_{\mathrm{pk}}(g^{f_j(a_{b_\Pi})};r(f_j(a_2)-f_j(a_1)))}$$
- $\mathcal{B}$ can compute encryption of $g^{f_j(a_{b_\Pi})}$ without knowing b_Π!

Proof: 2MHP are IND-CPA Secure V

$\mathcal{B}(a_1, a_2, \mathrm{pk}, c)$ $\mathcal{A}(a_1, a_2)$

For $j \in \{1, \ldots, t\}$:
$c_j \leftarrow (E_{\mathrm{pk}}(g^2; 0)/c)^{f_j(a_1)} \cdot (c/E_{\mathrm{pk}}(g; 0))^{f_j(a_2)} \cdot E_{\mathrm{pk}}(1; \mathcal{R})$

$\xrightarrow{(\mathrm{pk}; c_1, \ldots, c_t)}$

$b'_{\Gamma \leftarrow \mathcal{A}(\mathrm{pk}; c_1, \ldots, c_t)}$

$\xleftarrow{b'_\Gamma}$

$b'_{\Pi \leftarrow b'_\Gamma}$

Proof: 2MHP are IND-CPA Secure VI

By previous discussion, $\mathcal{B}$'s input to Γ is equal to his honest input corresponding to a_{b_Π} even if he does not know b_Π.

Assume $\mathcal{A}$ is successful with probability ε. Then $\mathcal{B}$ is successful with probability

$$\Pr[b'_\Pi = b_\Pi] = \Pr[b'_\Gamma = b_\Gamma] = \varepsilon \enspace .$$

$\mathcal{B}$'s time is dominated by the execution of $\mathcal{A}$ and $2t$ exponentiations. QED

Conclusions

- All homomorphic protocols are IND-CPA secure given PKC is IND-CPA secure
- We can always cite this metatheorem!
  - E.g.: if PKC is IND-CPA secure, then Hamming distance protocol is IND-CPA secure
- No significant security loss in ε or τ
  - Surprising: we intuitively expect that since attacker of Γ sees more than 1 ciphertext, he gains more advantage than when seeing just one
- Proof uses same homomorphic properties of Π
- We will deal with server's security later

Different Homomorphism: E-Voting I

- Two candidates, 0, 1
- Assume voter v_i, $i \in \{1, \dots, V\}$, votes for candidate $c_i \in \{0, 1\}$
- Voter v_i encrypts his ballot as $C_i \leftarrow E_{\mathrm{pk}}(g^{c_i}; r_i)$, sends it to vote collector
- At the end, vote collector "sums" all ballots as $C \leftarrow \prod_{i=1}^{V} C_i = E_{\mathrm{pk}}(g^{\sum_{i=1}^{V} c_i}; \sum_{i=1}^{V} r_i)$ $= E_{\mathrm{pk}}(g^{|\{i : c_i = 1\}|}; \sum_{i=1}^{V} r_i)$

Different Homomorphism: E-Voting II

- Vote collector does not know sk, it is only known by separate tallier
- Vote collector sends $C \cdot E_{\mathsf{pk}}(1; \mathcal{R})$ to tallier
- By decrypting the result and taking discrete logarithm of it, tallier finds $|\{i : c_i = 1\}|$, and declares 1 as winner exactly if that value is $> 50\%$ of voters
- Computation is efficient if number of voters is "small"
    - DL of number from $\{0, \dots, 2^n - 1\}$ can be done in time $2^{n/2} = \sqrt{2^n}$ by standard algorithms

Different Homomorphism: E-Voting III

- Viable say for $n \leq 80$ — and number of voters is smaller than 2^{80}!
- World population: $< 2^{33}$

Multiple-Candidate Elections I

- γ candidates mapped to $\{0, \ldots, \gamma - 1\}$
- Voter v_i prefers candidate c_i. His ballot is $C_i \leftarrow E_{\mathrm{pk}}(g^{(V+1)^{c_i}}; r_i)$
- Denote $T_k = |\{i : c_i = k\}|$ — number of voters who voted for k
- “Sum”: $\prod_{i=1}^{V} C_i = E_{\mathrm{pk}}(g^{\sum_{i=1}^{V}(V+1)^{c_i}}; \sum_{i=1}^{V} r_i)$
- Intuition:
  - All voters who vote for k contribute g^{V^k} to sum
  - Thus sum is $g^{\sum_{i=0}^{\gamma-1} T_i \cdot (V+1)^i}$

Multiple-Candidate Elections II

- Basis $V + 1$ was chosen here so that there are no overflows: $T_i < V + 1$ and thus $T_i(V + 1)^i < (V + 1)^{i+1}$
- Tallier takes discrete logarithm of sum, obtains $\sum_{i=0}^{\gamma-1} T_i(V + 1)^i$
- Tallier looks at this as number in $(V + 1)$-ary number system, where ith "digit" is equal to T_i
- Tallier extracts all digits $(T_0, \ldots, T_{\gamma-1})$

See [Cramer et al., 1997, Damgård and Jurik, 2001]

Problems with MC Elections

- Maximum value for "sum" may be just slightly smaller than $g^{(V+1)^\gamma}$
- Assume $V = 2^{20} - 1$ (appr million), $\gamma = 2^3 = 8$ (usual Estonian parliamentary election, voting for parties)
- $g^{(V+1)^\gamma} = g^{160}$, and computing DLs of this (2^{80} steps) is intractable!

Fourth Lecture: Additively Homomorphic Encryption

What Went Wrong?

- We always utilized multiplicatively homomorphic PKC (Elgamal) as additively homomorphic PKC in exponents, but at the end, one party had to compute DL
- By assumption if MH PKC, then DL is hard!
- Thus MH PKC is mostly only useful for applications where the final result comes from small (or well-structured) set

Lifted Elgamal

- Define lifted Elgamal (G, E, D) as follows
- Let $\mathbb{G}$ be cyclic multiplicative group of prime order q, generator $g \in \mathbb{G}$
- **Key generation:** choose $\mathsf{sk} \leftarrow \mathbb{Z}_q$, $\mathsf{pk} = h \leftarrow g^{\mathsf{sk}}$
- **Encryption:** set $r \leftarrow \mathbb{Z}_q$, $c = (c_1, c_2) = E_{\mathsf{pk}}(m; r) := (g^m h^r, g^r)$
- **Decryption:** set $D_{\mathsf{pk}}(c) = \log_g(c_1/c_2^{\mathsf{sk}})$
- Correctness: $D_{\mathsf{pk}}(E_{\mathsf{pk}}(m; r)) = \log_g(g^m h^r/(g^r)^{\mathsf{sk}}) = \log_g g^m = m$

Lifted Elgamal

- Additive homomorphism:
  $E_{\mathsf{pk}}(m_1; r_1) \cdot E_{\mathsf{pk}}(m_2; r_2) = (g^{m_1+m_2} h^{r_1+r_2}, g^{r_1+r_2})$
  $= E_{\mathsf{pk}}(m_1 + m_2; r_1 + r_2)$
- All previous protocols can be rewritten in terms of lifted Elgamal, with small modifications
  - $E_{\mathsf{pk}}(g^a; r) \to E_{\mathsf{pk}}(a; r)$ and
    $E_{\mathsf{pk}}(a; r) \to E_{\mathsf{pk}}(\log_g a; r)$
  - $\log_g D_{\mathsf{sk}}(c) \to D_{\mathsf{sk}}(c)$ and $D_{\mathsf{sk}}(c) \to g^{D_{\mathsf{sk}}(c)}$
- All previous protocols and security results work
- Decryption is inefficient unless in a small plaintext space

Hamming Distance with Lifted Elgamal

Alice $(a_1, \dots, a_t)$

Bob $(b_1, \dots, b_t)$

Alice: $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$, $(r_1, \dots, r_t) \leftarrow \mathcal{R}^t$, $c_i \leftarrow E_{pk}(a_i; r_i)$

Alice → Bob: $(\mathsf{pk}, (c_1, \dots, c_t))$

Bob: $r \leftarrow \mathcal{R}$, $c \leftarrow \prod_{i=1}^t (E_{\mathsf{pk}}(b_i; 0) \cdot c_i^{(-1)^{b_i}}) \cdot E_{\mathsf{pk}}(0; r)$

Bob → Alice: c

Alice: $m \leftarrow D_{sk}(c)$

Efficiency

- While efficiency of cryptographic protocols is very important, we have not talked about it much
- Several measures:
  - Communication complexity
  - Computational complexity (of Alice/Bob)
  - Round complexity
- Up to now all protocols have had 2 messages

Efficiency of HD Protocol with L. Elgamal

- Communication complexity: 1 PK + t ciphertexts $= 2t+1$ group elements
- 1 elliptic curve group element is 160 bits, thus $320t+160$ bits
- Alice's computation (dominated by): t enc + 1 dec $= 2t+1$ exp + 1 DL
- Bob's computation (dom by): $\leq t$ inversions ($\approx t$ mults) and $t+1$ mult
- $E_{\mathrm{pk}}(b_i; 0) = (g^{b_i}, g)$ can be precomputed for $b_i \in \{0, 1\}$ (costless — no exps)
- $E_{\mathrm{pk}}(0; r) = (h^r, g^r)$ (2 exps)
- $c_i^{(-1)^{b_i}}$ is either c_i or c_i^{-1} (no exp)
- 1 exp $\approx 1.5 \log q = 240$ mults, 1 DL $\approx 2^{t/2}$ mults
- Alice: $\approx 480t + 120 + 2^{t/2}$ mults
- DL time dominates for $t \geq 28$
- Bob: $\leq 2t+1$ mults

Alice $(a_1, \ldots, a_t)$		Bob $(b_1, \ldots, b_t)$
$(\mathrm{sk}, \mathrm{pk}) \leftarrow G(1^\kappa)$, $(r_1, \ldots, r_t) \leftarrow \mathcal{R}^t$, $c_i \leftarrow E_{pk}(a_i; r_i)$		
	$\xrightarrow{(\mathrm{pk}, (c_1, \ldots, c_t))}$	
		$r \leftarrow \mathcal{R}$, $c \leftarrow \prod_{i=1}^t (E_{\mathrm{pk}}(b_i; 0) \cdot c_i^{(-1)^{b_i}}) \cdot E_{\mathrm{pk}}(0; r)$
	$\xleftarrow{c}$	
$m \leftarrow D_{sk}(c)$		

Efficiency w (L.) Elgamal: General

- Alice:
  - To encrypt t plaintexts, Alice encrypts t times — $2t$ exp $= 3t \log q$ mults
  - Alice decrypts/computes DL say s times - $s(1.5 \log q + 2^{n/2})$ mults for some n
  - Total: $3t \log q + s(1.5 \log q + 2^{n/2})$ mults
  - Plus may be some additional ops
  - Inherit lower bound
  - Goal of protocol designer is to minimize t, s and n
- Bob's efficiency can vary

Additively Homomorphic Cryptosystems

- PKC (G, E, D) with
  $E_{pk}(m_1; r_1) \cdot E_{pk}(m_2; r_2) = E_{pk}(m_1 + m_2; r_1 \circ r_2)$
- With efficient decryption — no need to compute DL!
- Lifted Elgamal: AH for small plaintext group
- Need AH PKC with large plaintext group
  - Paillier [Paillier, 1999]: $\mathbb{Z}_n$ with $n > 2^{1536}$
  - Damgård-Jurik [Damgård and Jurik, 2001]: $\mathbb{Z}_n^s$ with $n > 2^{1536}$ and integer $s \geq 1$

Background: Factoring Assumption

Let $\ell = \ell(\kappa)$ some bitlength, and $\mathcal{A} = \mathcal{A}_\ell$ be a non-uniform adversary. Let $\mathfrak{P}_\ell$ be the set of all ℓ-bit primes. Define

$$Adv_\ell^{fact}(\mathcal{A}) := \Pr[p, q \leftarrow \mathfrak{P}_\ell, n \leftarrow p \cdot q : \mathcal{A}(n) = (p, q)]$$

Factoring 2ℓ-bit RSA moduli is hard if for any non-uniform probabilistic adversary $\mathcal{A} = \mathcal{A}_\ell$ that works in time $\leq \tau$, $Adv_\ell^{fact}(\mathcal{A}) \leq \varepsilon$.
Best factorization algorithm (GNFS) works in time $e^{(\sqrt[3]{64/9}+o(1))(\log n)^{1/3}(\log \log n)^{2/3}}$ for integer n

Corollaries of Factoring Assumption I

- If factoring is hard, then computing $\varphi(n)$ for random RSA modulus n is hard
  - $\varphi(n) = \varphi(pq) = (p-1)(q-1) = pq - p - q + 1$
  - If one knows both n and $\varphi(n)$, one also knows $s = n - \varphi(n) + 1 = p + q$
  - $n = pq = p(s-p) = sp - p^2$, thus $p^2 - sp + n = 0$ — quadratic equation
  - One can recover $p \leftarrow (s \pm \sqrt{s^2 - 4n})/2$
  - Example: $n = 4347803203$, $\varphi(n) = 4347671328$
  - Thus $s = 131876$, and $p = 65809$ or $p = 66067$. In fact, $65809 \cdot 66067 = 4347803203$

Corollaries of Factoring Assumption II

- Since $\phi(n) = |\mathbb{Z}_n^*|$, if $y = x^e \mod n$ then $x = y^{e^{-1} \mod \phi(n)} \mod n$. Finding $e^{-1} \mod \phi(n)$ is hard without knowing how to factor n
- A lot of other things are hard if factoring is hard

Background: Binomial Theorem and DL

- $(a+b)^c = \sum_{i=0}^{c} \binom{c}{i} a^i b^{c-i}$
- For example:
  - $(n+1)^c = \sum_{i=0}^{c} \binom{c}{i} n^i =$
    $1 + cn + \binom{c}{2} n^2 + \text{higher powers of } n$
  - $(n+1)^c \equiv cn + 1 \pmod{n^2}$
- Can compute certain discrete logarithms easily:
  - If $y = (n+1)^x \mod n^2$, then $y = xn + 1 \mod n^2$
  - Thus $x = (y-1)/n \mod n^2$
- Denote $L(y) := \frac{y-1}{n}$ (quotient of integer division)
- Thus: $L((n+1)^x \mod n^2) = x$

Background: Basic Number Theory

- $\mathrm{lcm}(a, b)$ — least common multiplier
  - $a \mid \mathrm{lcm}(a, b)$, $b \mid \mathrm{lcm}(a, b)$
  - If $a \mid c$ and $b \mid c$, then $b \leq c$
- $a \cdot b = \gcd(a, b) \cdot \mathrm{lcm}(a, b)$
  - Example: $a = 4$, $b = 6$
  - $\gcd(4, 6) = 2$, $\mathrm{lcm}(4, 6) = 12$
  - $4 \cdot 6 = 24 = 2 \cdot 12$

Background: Carmichael Function

- **Def:** for positive integer n, smallest positive integer $\lambda(n) = m$ such that $a^m \equiv 1 \pmod{n}$ for every integer a coprime to n.
- $\lambda(p^k) = p^{k-1}(p-1)$ if $p \geq 3$ or $k \leq 2$ $(= \varphi(p^k))$,
  $\lambda(2^k) = 2^{k-2}$ for $k \geq 3$, and
  $\lambda(p_1^{k_1} \dots p_t^{k_t}) = \mathrm{lcm}(\lambda(p_1^{k_1}), \dots, \lambda(p_t^{k_t}))$

Theorem (Carmichael Theorem)

If $\gcd(a, n) = 1$ *then* $a^{\lambda(n)} \equiv 1 \pmod{n}$.

Full proof is 6+ pages.

Paillier's Cryptosystem: Key Generation

- Generate two independent random large prime numbers p and q // both ≥ 768 bits
- Let $n \leftarrow p \cdot q$
- Let $\lambda \leftarrow \lambda(n) = \mathrm{lcm}(p-1, q-1)$
- Let $\mu \leftarrow \lambda^{-1} \mod n$.
- The public key is $\mathsf{pk} = n$, the private key is $\mathsf{sk} = (\lambda, \mu)$

Paillier's Cryptosystem

- **Encryption of $m \in \mathbb{Z}_n$ with $\mathsf{pk} = n$:**
  Select random $r \leftarrow \mathbb{Z}_n^*$. Compute

$$c \leftarrow (n+1)^m r^n \mod n^2$$

  Note: $c = (mn+1)r^n \mod n^2$
  r has order $\varphi(n) = (p-1)(q-1)$.
- **Decryption of $c \in \mathbb{Z}_{n^2}^*$ with $\mathsf{sk} = (\lambda, \mu)$:**

$$m \leftarrow L(c^\lambda \mod n^2) \cdot \mu \mod n$$

Correctness of Paillier Decryption

For $\mathsf{sk} = (\lambda, \mu)$ and $\mathsf{pk} = n$,

$$\begin{aligned} D_{\mathsf{sk}}(E_{\mathsf{pk}}(m; r)) &\equiv D_{\mathsf{sk}}((n+1)^m r^n \mod n^2) \\ &\equiv L((n+1)^{\lambda m} r^{\lambda n} \mod n^2) \cdot \mu \\ &\equiv L((\lambda m n + 1) r^{\lambda n} \mod n^2) \cdot \mu \pmod n \ . \end{aligned}$$

We have to get rid of $r^{\lambda n}$

Correctness of Paillier Decryption

Now, $\lambda(n^2) = \lambda(p^2q^2) = \mathrm{lcm}(\lambda(p^2), \lambda(q^2)) = \mathrm{lcm}(p(p-1), q(q-1)) = pq \cdot \mathrm{lcm}(p-1, q-1) = \lambda n$.
By Carmichael theorem, $r^{\lambda n} \equiv r^{\lambda(n^2)} \equiv 1 \mod n^2$.
Thus

$$\begin{aligned} D_{\mathsf{sk}}(E_{\mathsf{pk}}(m; r)) &\equiv L(\lambda mn + 1) \cdot \mu \\ &\equiv \lambda m \cdot \lambda^{-1} \\ &\equiv \frac{\lambda m}{\lambda} \equiv m \pmod{n} . \end{aligned}$$

Paillier: Homomorphism

Clearly,

$$\begin{aligned}E_{\mathrm{pk}}(m_1; r_1) \cdot E_{\mathrm{pk}}(m_2; r_2) &\equiv (n+1)^{m_1} {r_1}^n \cdot (n+1)^{m_2} \cdot {r_2}^n \\ &\equiv (n+1)^{m_1+m_2} (r_1 r_2)^n \\ &\equiv E_{\mathrm{pk}}(m_1 + m_2; r_1 \cdot r_2) \pmod{n^2}\end{aligned}$$

Thus the Paillier cryptosystem is homomorphic in $\mathcal{M} = \mathbb{Z}_n$.

Security of Paillier

x is n-th residue modulo n^2 iff there exists y such that $y^n \equiv x \pmod{n^2}$

Definition

Decisional Composite Residuosity Assumption: Distinguish a random n-th residue from a random n-th non-residue modulo n^2.

Equivalent (with small error): Distinguish a random n-th residue from a random element of $\mathcal{C} = \mathbb{Z}_{n^2}$.

Fact: If factoring is easy, then DCRA is easy.

Opposite is not known.

Security of Paillier

Theorem

Assume that DCRA is true. Then Paillier is IND-CPA secure.

Sketch.

Idea: random encryption of 0 is a random n-th residue; random encryption of a random element in $\mathcal{M}$ is a random element of $\mathcal{C}$. Proof goes along the same lines as the security proof of Elgamal. $\square$

Efficiency of Paillier

- $\log n \geq 1536$ (need hardness of factoring)
- Encryption: dom. by 1 1536-bit exp — $\approx$ 2304 3072-bit multiplications
  - Less efficient than lifted Elgamal on elliptic curve groups (10x more mults, bitlength 20x longer)
- Decryption: dom. by 1 3072-bit exp — $\approx$ 2304 3072-bit multiplications
  - Significantly more efficient than lifted Elgamal: polynomial instead of exponential — thus can decrypt much larger plaintexts
- Ciphertext: 3072 bits

2-Message AH Protocols

- a — anything (e.g., a real value)
- $m_i \in \mathcal{M}$ are functions of a
- $m_i = m_i(a)$

Except this sentence, this is copy of previous slide!

Alice (a) — Bob (b)

$(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$,
For $i \in \{1, \dots, t\}$,
$c_i \leftarrow E_{\mathsf{pk}}(m_i, r_i)$

$$\xrightarrow{(\mathsf{pk}; c_1, \dots, c_t)}$$

$\mathfrak{r} \leftarrow \mathit{Reply}(1^\kappa, b, \mathsf{pk}, c_1, \dots, c_t)$

$$\xleftrightarrow{\mathfrak{r}}$$

$\mathfrak{a} = \mathit{Answer}(1^\kappa, a, \mathsf{sk}, \mathsf{pk}, \mathfrak{r})$

Efficiency of HD Protocol with Paillier

- Communication complexity: 1 PK $+$ t ciphertexts $=$ n and $2t$ integers modulo n^2
- $1536 + 6144t$ bits
- Alice's computation (dominated by): t enc $+$ 1 dec $=$ $t+1$ exp
- Bob's computation (dom by): $\leq t$ inversions ($\approx t$ mults) and $t+1$ mult
- 1 exp $\approx 1.5 \log n = 2304$ mults
- Alice: $\approx 2304t + 2304$ mults
- Bob: $\approx 2t + 1$ mults
- Here: 3072-bit mult, in Elgamal – 160-bit mult (much faster)

Alice $(a_1, \ldots, a_t)$ | Bob $(b_1, \ldots, b_t)$

Alice: $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$, $(r_1, \ldots, r_t) \leftarrow \mathcal{R}^t$, $c_i \leftarrow E_{pk}(a_i; r_i)$

$$\xrightarrow{(\mathsf{pk}, (c_1, \ldots, c_t))}$$

Bob: $r \leftarrow \mathcal{R}$, $c \leftarrow \prod_{i=1}^{t} (E_{\mathsf{pk}}(b_i; 0) \cdot c_i^{(-1)^{b_i}}) \cdot E_{\mathsf{pk}}(0; r)$

$$\xleftarrow{c}$$

Alice: $m \leftarrow D_{sk}(c)$

Elgamal or Paillier

- If decrypted values not too big (DL efficient), use (lifted) Elgamal
- If decrypted values of average size, depends
  - Alice's ops are 10x faster but Bob's ops 50x slower — what is more important?
  - E.g.: homomorphic e-voting
- If decrypted values are large (DL intractable), use Paillier

Metatheorem: 2AHP are IND-CPA Secure

Theorem

Assume additively homomorphic $\Pi = (G, E, D)$ is IND-CPA secure. Then $\Gamma = (Query, Reply, Answer)$ is IND-CPA secure.

Proof.

Simple modification of MH case. Replace plaintexts g^x with plaintexts x. □

Fifth Lecture. Semisimulatability

For original definition of semisimulatability, see [Naor and Pinkas, 1999].
For our (me and Sven Laur) paper on DIE/CDS, see [Laur and Lipmaa, 2007]

Recap: 2-Message AH Protocols

- a — anything (e.g., a real value)
- $a_i(a) \in \mathcal{M}$ are functions of a
- Alice's privacy follows from IND-CPA of PKC

Alice (a) | Bob (b)

Alice: $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$,
For $i \in \{1, \dots, t\}$,
$c_i \leftarrow E_{\mathsf{pk}}(a_i, r_i)$

Alice → Bob: $(\mathsf{pk}; c_1, \dots, c_t)$

Bob: $\mathfrak{r} \leftarrow Reply(1^\kappa, b, \mathsf{pk}, c_1, \dots, c_t)$

Bob → Alice: $\mathfrak{r}$

Alice: $\mathfrak{a} = Answer(1^\kappa, a, \mathsf{sk}, \mathsf{pk}, \mathfrak{r})$

Recap: What Can Be Done with 2AH/2MH?

- Alice can encrypt arbitrary functions a_i of a
  - See m-c elections, Hamming distance protocol
- Bob can compute affine functions of encrypted values for some functions b_i, b' of b:

$$\text{MH: } \prod_i E_{\text{pk}}(g^{a_i}; r_i)^{b_i} \cdot E_{\text{pk}}(g^{b'}; r') = E_{\text{pk}}(g^{\sum_i b_i a_i + b'}; \cdot)$$

$$\text{AH: } \prod_i E_{\text{pk}}(a_i; r_i)^{b_i} \cdot E_{\text{pk}}(b'; r') = E_{\text{pk}}(\textstyle\sum_i b_i a_i + b'; \cdot)$$

- Quite limited — most freedom is in choosing a_i, b_i, b'

Can We Do More?

- Functionality:
  - Are there any non-algebraic things we can do?
  - More algebraic freedom — compute quadratic equations, ...?
  - Many rounds — will it help?
  - Many parties — will it help?
- Security:
  - Previous protocols guaranteed only Alice's privacy — can we do more?

This Lecture

- Functionality:
  - Are there any non-algebraic things we can do?
  - More algebraic freedom — compute quadratic equations, . . .?
  - Many rounds — will it help?
  - Many parties — will it help?
- Security:
  - Previous protocols guaranteed only Alice's privacy — can we do more?

Security in Malicious Model

- Alice:
  - Privacy: Bob does not learn Alice's input — IND-CPA security, we dealt with it
  - Security: Alice gets back correct answer — future lectures
- Bob:
  - Privacy: Alice does not learn more about Bob's input than necessary
  - Security: Bob gets back correct answer — easy

Recap: (Boolean) Scalar Product

- Alice has $(a_1, \ldots, a_t) \in \mathbb{Z}_2^t$
- Bob has $(b_1, \ldots, b_t) \in \mathbb{Z}_2^t$
- Alice learns $\sum_{i=1}^{t} a_i b_i \mod q \in \mathbb{Z}_q$
- Privacy in semihonest model:
  - Alice learns nothing else, Bob learns nothing
- What about privacy in malicious model?
  - Bob still learns nothing, what about Alice?

Within this lecture we use Elgamal & corresponding notation

Cheating the Scalar Product

- Alice obtains $\sum_{i=1}^{t} a_i b_i \mod q$
- Malicious Alice sets $a_i \leftarrow 2^i$
- $\sum_{i=1}^{t} a_i b_i = \sum_{i=1}^{t} 2^i b_i \mod q$
- Alice recovers Bob's whole input!

Alice $(a_1, \ldots, a_t) \in \mathbb{Z}_2^t$ | Bob $(b_1, \ldots, b_t) \in \mathbb{Z}_2^t$

Alice: $(\mathrm{sk}, \mathrm{pk}) \leftarrow G(1^\kappa)$, $(r_1, \ldots, r_t) \leftarrow \mathcal{R}^t$, $c_i \leftarrow E_{pk}(g^{a_i}; r_i)$

Alice → Bob: $(\mathrm{pk}, (c_1, \ldots, c_t))$

Bob: $r \leftarrow \mathcal{R}$, $c \leftarrow \prod_{i=1}^{t} c_i^{b_i} \cdot E_{\mathrm{pk}}(1; r)$

Bob → Alice: c

Alice: $m \leftarrow \log_g D_{sk}(c)$

Getting Bob's Privacy. First Idea

- Malicious Alice can only attack SSP by encrypting values out of range
- Make it so that if Alice encrypts wrong values then Alice gets back garbage!

Randomizing Elgamal Plaintexts

- Plaintext group $\mathcal{M}$ is cyclic of prime order q. Let g be generator
- For fixed $y = g^x \in \mathcal{M}$, and random $r \leftarrow \mathbb{Z}_q$,

$$y^r = g^{xr} = \begin{cases} g \ , & x = 0 \ , \\ \text{random element of } \mathbb{G} \ , & \text{otherwise} \ . \end{cases}$$

- Latter holds since if $x \neq 0$ and r is random, then $xr \mod q$ is a random element of $\mathbb{Z}_q$
- Thus $E_{\mathsf{pk}}(m; s)^r$ for random r encrypts 1 if $m = 1$, and encrypts random plaintext if $m \neq 1$

More Than Just Algebra

- Alice can encrypt arbitrary functions a_i of a
  - See multi-candidate elections, Hamming distance protocols
- Bob can compute affine functions of encrypted values, $\prod_i E_{\mathrm{pk}}(g^{a_i}; r_i)^{b_i} \cdot E_{\mathrm{pk}}(g^{b'}; \mathcal{R}) = E_{\mathrm{pk}}(g^{\sum_i b_i a_i + b'}; \mathcal{R})$
- Bob can conditionally randomize plaint-s: $(\prod_i E_{\mathrm{pk}}(g^{a_i}; r_i)^{b_i} \cdot E_{\mathrm{pk}}(g^{b'}; 0))^{\mathbb{Z}_q} \cdot E_{\mathrm{pk}}(g^{b''}; \mathcal{R})$ encrypts $g^{b''}$ if $\sum_i b_i a_i + b' = 0$, and a random value otherwise

Disclose-if-Equal Protocol with Elgamal

- Alice's input is $a \in \mathbb{Z}_q$
- Bob's input is $b \in \mathbb{Z}_q$, $b' \in \mathcal{M}$
- Alice obtains b' if $a = b$ and random value if $a \neq b$
- Note: one could also choose $a, b \in \mathbb{G}$
    - In this application, using MH cryptosystem does not mean that one has to compute discrete logarithm!
    - However since we use DIE mostly to secure other protocols, we use g^a/g^b instead of a/b
    - We however use $b' \in \mathcal{M}$

Disclose-if-Equal Protocol with Elgamal

Alice $a \in \mathbb{Z}_q$

Bob $b \in \mathbb{Z}_q, b' \in \mathcal{M}$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$,
$r_a \leftarrow \mathcal{R}$,
$c \leftarrow E_{pk}(g^a; r_a)$

$\xrightarrow{(\mathsf{pk}, c)}$

$r_b \leftarrow \mathbb{Z}_q$, $r'_b \leftarrow \mathcal{R}$,
$c' \leftarrow (c \cdot E_{\mathsf{pk}}(g^{-b}; 0))^{r_b} \cdot E_{\mathsf{pk}}(b'; r'_b)$

$\xleftarrow{c'}$

$m \leftarrow D_{sk}(c')$ // No DL!

Correctness of DIE Protocol

Recall $c = E_{\mathrm{pk}}(g^a; r_a)$. Then

$$c' = \underbrace{\underbrace{\underbrace{(c \cdot E_{\mathrm{pk}}(g^{-b}; 0))}_{E_{\mathrm{pk}}(g^{a-b}; r_a)}{}^{r_b}}_{E_{\mathrm{pk}}(g^{(a-b)r_b}; r_a r_b)} \cdot E_{\mathrm{pk}}(b'; r'_b)}_{E_{\mathrm{pk}}(g^{(a-b)r_b} \cdot b'; r_a r_b + r'_b)}$$

Since $r'_b \leftarrow \mathbb{Z}_q$ is random, c' is random encryption of $g^{(a-b)r_b} \cdot b'$. Since r_b is random, then $D_{\mathrm{sk}}(c') = b'$ if $a = b$ and random if $a \neq b$.

Bob's Privacy in DIE

- As we showed, Alice obtains random encryption of b' if $a = b$ and random encryption of random plaintext if $a \neq b$
- The latter contains no information about b
- Intuitively, thus the protocol is private for Bob
- How to formalize?

Simulation I

- Want: Bob's second message $\mathfrak{r}$ gives Alice no extra information compared to what she would have given her input a, first message $\mathfrak{q}$, and rightful output $\mathfrak{a} = f(a, b)$ of protocol
  - Instead of a we take a^*, set of plaintexts encrypted by Alice in $\mathfrak{q}$
  - Reasoning: malicious Alice has no well-defined input. It only matters what she did send to Bob
- If Alice can construct $\mathfrak{r}$ herself, given $(a, \mathfrak{q}, \mathfrak{a})$, she gains no more information from $\mathfrak{r}$

Simulation II

- We construct simulator that, given $(a, \mathfrak{q}, \mathfrak{a})$, constructs simulated second message $\mathfrak{r}^*$
- Required: $(a, \mathfrak{q}, \mathfrak{r}, \mathfrak{a})$ and $(a, \mathfrak{q}, \mathfrak{r}^*, \mathfrak{a})$ are indistinguishable — come from (almost) same distributions

Recap: DIE Protocol

- Input a^* ($= g^a$ if Alice is honest)
- $\mathfrak{a} = b'$ if $a^* = g^b$,
  $\mathfrak{a} = \mathcal{M}$ if $a^* \neq g^b$
- $\mathfrak{q} = (\mathsf{pk}, c)$
- $\mathfrak{r} = (c' = E_{\mathsf{pk}}(\mathfrak{a}; \mathcal{R}))$

Alice $a \in \mathbb{Z}_q$ | Bob $b \in \mathbb{Z}_q, b' \in \mathcal{M}$

Alice: $(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$, $r_a \leftarrow \mathcal{R}$, $c \leftarrow E_{pk}(g^a; r)$

$$\xrightarrow{(\mathsf{pk}, c)}$$

Bob: $r_b \leftarrow \mathbb{Z}_q$, $r'_b \leftarrow \mathcal{R}$, $c' \leftarrow (c \cdot E_{\mathsf{pk}}(g^{-b}; 0))^{r_b} \cdot E_{\mathsf{pk}}(b'; r'_b)$

$$\xleftarrow{c'}$$

Alice: $\mathfrak{a} \leftarrow D_{sk}(c)$

Simulator for DIE Protocol

- Simulator gets $(a^*, \mathfrak{q} = (\mathsf{pk}, c), \mathfrak{a})$ where

$$\mathfrak{a} = \begin{cases} b', & a^* = g^b, \\ \mathcal{M}, & a^* \neq g^b. \end{cases}$$

- Simulator returns

$$\mathfrak{r}^* := E_{\mathsf{pk}}(\mathfrak{a}; \mathcal{R}) = \begin{cases} E_{\mathsf{pk}}(b'; \mathcal{R}), & a^* = g^b, \\ E_{\mathsf{pk}}(\mathcal{M}; \mathcal{R}), & a^* \neq g^b. \end{cases}$$

without knowing (b, b')

- Clearly $\mathfrak{r}^* = \mathfrak{r}$ as a distribution

Semisimulatability

- 2-message protocol is semisimulatable if:
  - Alice's privacy is guaranteed by IND-CPA security
  - Bob's privacy is guaranteed by above definition of simulatibility
- Simulatability is stronger than IND-CPA security
  - It expresses what we want from protocol
  - Simulatable protocols are usually much less efficient
- Fully simulatable security — future lectures

Terminology: Semisimulatable = half-simulatable = relaxed secure

DIE Protocol Is Semisimulatable

Theorem

DIE protocol is semisimulatable.

Proof.

IND-CPA security follows from earlier metatheorem.
We just showed Bob's privacy. $\square$

Constructing Semisimulatable Protocols

- Construct 2-message homomorphic protocol
- Make it Bob-private by using CDS — suitable generalization of DIE protocol
- Conditional Disclosure of Secrets: Alice obtains Bob's answer iff Alice's encrypted inputs belong to some public set $\mathcal{S}$ of valid inputs. Otherwise Alice obtains random value [Aiello et al., 2001, Laur and Lipmaa, 2007]

Reminder: Scalar Product Protocol

- Alice obtains $\sum_{i=1}^{t} a_i b_i \mod q$
- Valid inputs: $a_i \in \{0, 1\}$ for $t \in \{1, \dots, t\}$
- Boolean formula for valid inputs: $\bigwedge_{i=1}^{t} (a_i = 0 \vee a_i = 1)$

Alice $(a_1, \dots, a_t) \in \mathbb{Z}_2^t$ | Bob $(b_1, \dots, b_t) \in \mathbb{Z}_2^t$

Alice: $(\mathrm{sk}, \mathrm{pk}) \leftarrow G(1^\kappa)$, $(r_1, \dots, r_t) \leftarrow \mathcal{R}^t$, $c_i \leftarrow E_{pk}(g^{a_i}; r_i)$

$\xrightarrow{(\mathrm{pk}, (c_1, \dots, c_t))}$

Bob: $r \leftarrow \mathcal{R}$, $c \leftarrow \prod_{i=1}^{t} c_i^{b_i} \cdot E_{\mathrm{pk}}(1; r)$

$\xleftarrow{c}$

Alice: $m \leftarrow \log_g D_{sk}(c)$

Semisim. SSP: Idea

- Idea:
  - Alice obtains secret s_i if $a_i = 0$ or $a_i = 1$
  - Alice obtains $s = \sum_{i=1}^{t} s_i$ if he knows all values s_i
  - Alice obtains $\sum a_i b_i + s$. Thus Alice obtains $\sum a_i b_i$ only if $a_i \in \{0, 1\}$ for all i

Semisimulatable SSP

Alice $(a_1, \ldots, a_t) \in \mathbb{Z}_2^t$

Bob $(b_1, \ldots, b_t) \in \mathbb{Z}_2^t$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow G(1^\kappa)$,
$(r_1, \ldots, r_t) \leftarrow \mathcal{R}^t$,
$c_i \leftarrow E_{pk}(g^{a_i}; r_i)$

$\mathfrak{q} \leftarrow (\mathsf{pk}, (c_1, \ldots, c_t))$ $\longrightarrow$

If $\mathfrak{q} \notin \mathbb{G}^{2t+1}$, then halt.
$r, s_1, \ldots, s_t, \left(r'_{ij}, r''_{ij}\right)_{i \in \{1,\ldots,t\}, j \in \{0,1\}} \leftarrow \mathbb{Z}_q$,
For $i \in \{1, \ldots, t\}$ and $j \in \{0, 1\}$
$c'_{ij} \leftarrow \left(c_i / E_{\mathsf{pk}}(g^j; 0)\right)^{r'_{ij}} \cdot E_{\mathsf{pk}}(g^{s_i}; r''_{ij})$
$c \leftarrow \prod_{i=1}^t c_i^{b_i} \cdot E_{\mathsf{pk}}\left(g^{\sum_{i=1}^t s_i}; r\right)$

$\longleftarrow$ $\mathfrak{r} \leftarrow ((c'_{ij})_{i \in \{1,\ldots,t\}, j \in \{0,1\}}, c)$

For $i \in \{1, \ldots, t\}$: $w_i \leftarrow D_{\mathsf{sk}}(c'_{i,a_i})$
$\mathfrak{a} \leftarrow \log_g(D_{sk}(c) / \prod_{i=1}^t w_i)$

Semisimulatable SSP: Correctness I

Recall $c_i = E_{\mathrm{pk}}(g^{a_i}; r_i)$ for some a_i, r_i. Then $c = \prod_{i=1}^{t} E_{\mathrm{pk}}(g^{a_i}; r_i)^{b_i} \cdot E_{\mathrm{pk}}(g^{\sum_{i=1}^{t} s_i}; r) = E_{\mathrm{pk}}(g^{\sum_{i=1}^{t} a_i b_i + \sum_{i=1}^{t} s_i}; \sum_{i=1}^{t} r_i b_i + r)$ and

$$c'_{ij} = \underbrace{\underbrace{\underbrace{(c_i / E_{\mathrm{pk}}(g^j; 0))}_{E_{\mathrm{pk}}(g^{a_i - j}; r_i)}{}^{r'_{ij}}}_{E_{\mathrm{pk}}(g^{(a_i - j)\cdot r'_{ij}}; r_i r'_{ij})} \cdot E_{\mathrm{pk}}(g^{s_i}; r''_{ij})}_{E_{\mathrm{pk}}(g^{(a_i - j)\cdot r'_{ij} + s_i}; r_i r'_{ij} + r''_{ij})}$$

Semisimulatable SSP: Correctness II

Since r'_{ij}, r''_{ij} are random,

$$c'_{ij} = \begin{cases} E_{\mathrm{pk}}(g^{s_i}; \mathcal{R}) \ , & a_i = j \ , \\ E_{\mathrm{pk}}(\mathcal{M}; \mathcal{R}) \ , & a_i \neq j \ . \end{cases}$$

Thus $w_i \leftarrow g^{s_i}$, if Alice is honest. If Alice is malicious, $w_i \leftarrow \mathcal{M}$ (random). Thus if Alice is honest then $m = \log_2(g^{\sum a_i b_i}) = \sum a_i b_i$, otherwise $g^{\mathfrak{a}}$ is a random element of $\mathbb{G}$ (and computing DL is hard!)

Remarks: CDS with Paillier

- One can substitute Elgamal with Paillier, but it's more complex then
- $\mathcal{M} = \mathbb{Z}_n$ with $n = pq$ has nontrivial subgroups. If $a_i \neq 0$ belongs to some such subgroup $\mathcal{M}_1$, then $a_i \cdot \mathcal{M} = \mathcal{M}_1$, not $a_i \cdot \mathcal{M} = \mathcal{M}$
- If malicious Alice encrypts say p, then $D_{\mathsf{sk}}(E_{\mathsf{pk}}(p; \cdot)^{\mathcal{M}})$ divides by p and thus does not hide perfectly
- See [Laur and Lipmaa, 2007] for simple solution

Remarks

- One can generalize SSP example to CDS for arbitrary efficiently computable set $\mathcal{S}$
  - Write down circuit that computes $\mathcal{S}$. Handle AND/OR gates as in SSP case. For NOT gates, see [Laur and Lipmaa, 2007] (easy)
- **Example.** Assume that valid value of a_i is $a_i \in \{0, \ldots, 255\}$
  - Simplistic approach: distribute g^{s_i} iff $a_i = 0 \vee a_i = 1 \vee \cdots \vee a_i = 255$ — requires 256 ciphertexts
  - More efficient: encrypt bits a_{ij} of a_i separately. Distribute $g^{s_{ij}}$ if $a_{ij} = 0 \vee a_{ij} = 1$. Write $s_i = \sum_j s_{ij}$ — requires $2 \cdot 8 = 16$ ciphertexts