UNBOUND

[WHERE SECURITY IS KEY]

Introduction to MPC

Yehuda Lindell

CEO and co-Founder of Unbound Tech

Professor of Computer Science, Bar-Ilan University

What is MPC?

Secure Multiparty Computation

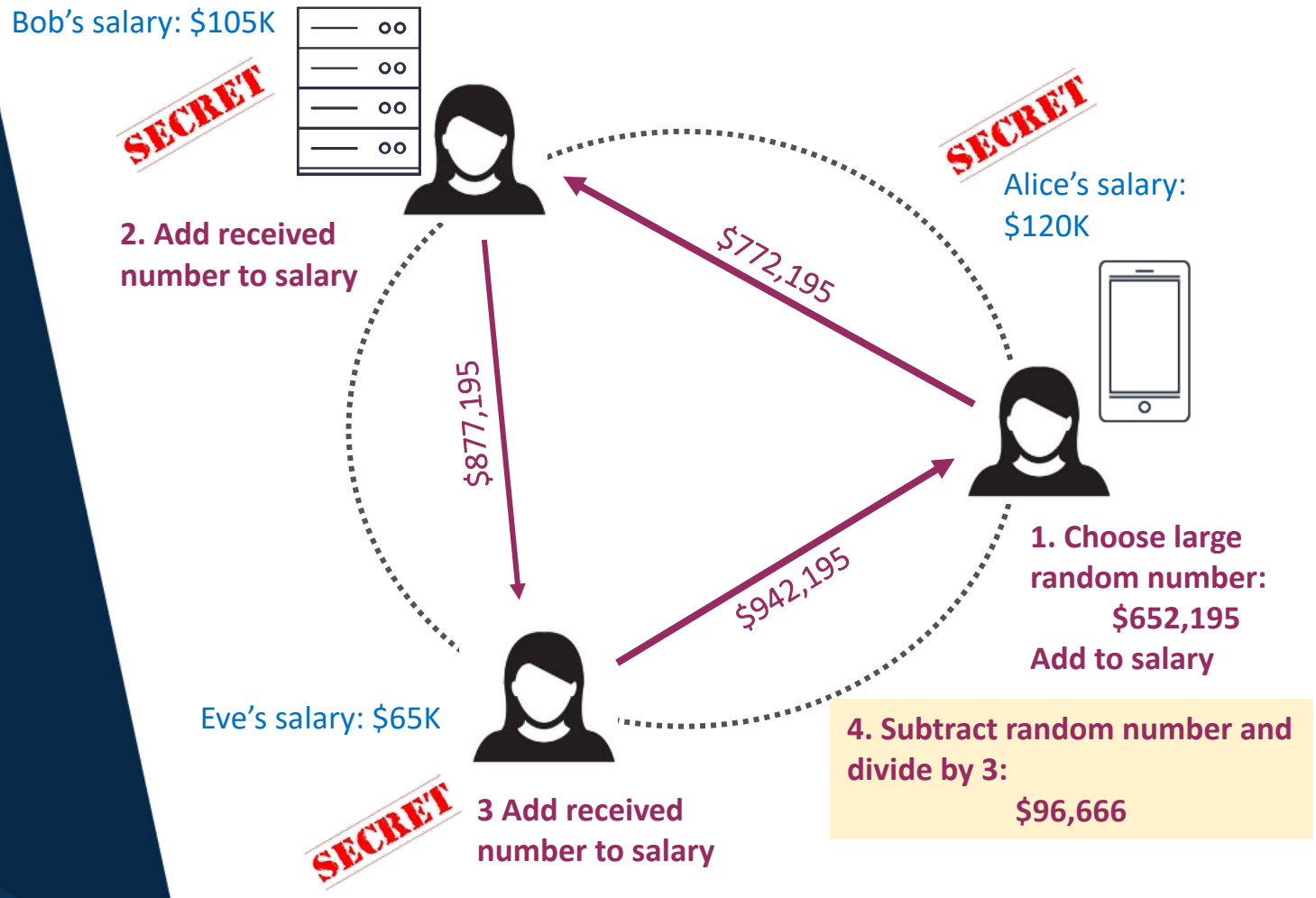
- **A well researched subfield of cryptography**
  - Research began in the late 1980s
  - Thousands of research papers
  - Research was purely theoretical until recently
  - MPC is now a **very active applied area of research**

- **The idea – compute on private data without revealing anything**

Secure Multiparty Computation Toy Example

Compute average salary:

A group of cryptographers want to compute the average of their salaries, without revealing anyone's salary!

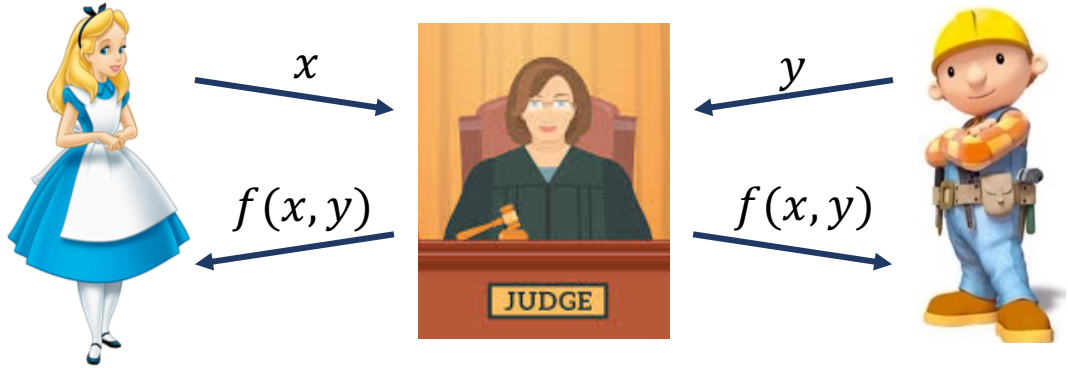


MPC Security Requirements

- **Parties with private inputs compute a joint function of their inputs**
  - Ensuring that **nothing but the output is learned** (privacy)
  - Ensuring that the **output is correctly computed** (correctness)
- **Properties should be guaranteed even in the face of adversarial behavior**
  - **Semi-honest**: adversary running the correct software cannot learn anything
  - **Malicious**: adversary running **any software** cannot learn anything
    - Even if they know all the protocols, design, and so on
- **The adversary can corrupt parties:**
  - Two main settings: any number (**dishonest majority**) or a minority (**honest majority**)
- **Security is mathematically proven**

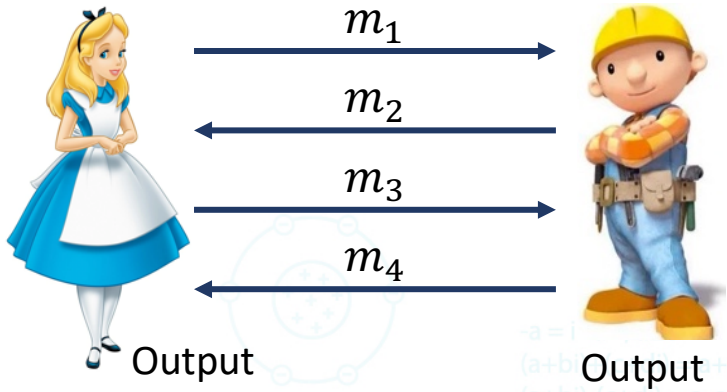
The Ideal/Real Paradigm

- **What would the ideal situation be?**
  - A trusted and incorruptible third party
  - All parties send inputs to trusted party
    - On perfectly-secure communication channels
  - Trusted party computes and sends output
- **Properties**
  - **Privacy**: each party learns nothing but their output
  - **Correctness**: output is correct
  - **More…**



The Ideal/Real Paradigm

- **The real world**
  - Parties interact with each other
  - There is no trusted party
  - Parties output what the protocol tells them to



- **Definition: an MPC protocol is secure if it "behaves like" an ideal world protocol**
  - Cannot do more than what an attacker can do in the ideal world
  - In the ideal world, can choose your input and that's it

Definitional Advantages

- **Very easy to understand – build and justify your application assuming a trusted party (secure black box)**
  - Don’t need to be a cryptographer!
- **Words of warning**
  - MPC talks about the process but not the function itself
    - Average of salaries between two people reveals everything in the ideal world
    - MPC of cryptographic functions is fine by definition, if cryptographic function is secure
  - Parties can choose their own inputs – if this is a problem, needs to be worked into the function definition

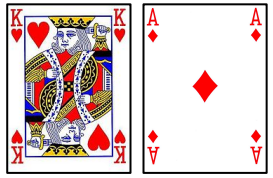
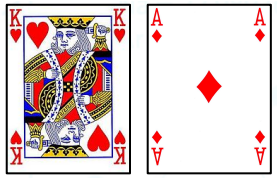
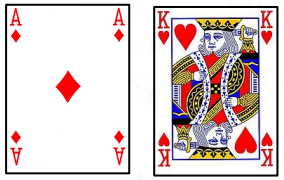
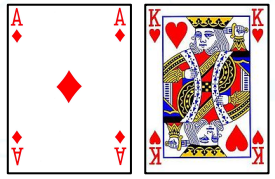
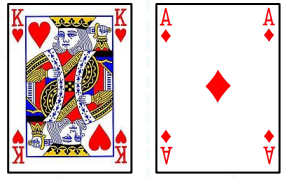
How Does MPC Work?

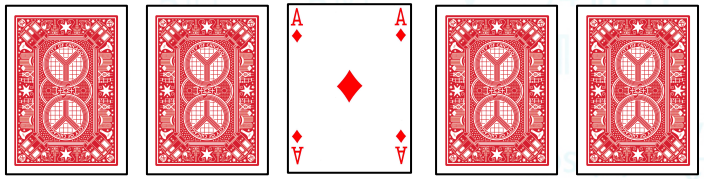
Secure Computation – A Fun Problem

- **Consider the dating problem**
  - A guy and a girl want to check if they are both interested in going out
  - If they both are, then output is YES
  - If at least one is not, then output is NO

- **If Alice says YES and Bob says NO, then the result is NO and Bob doesn't know if Alice said YES or not**
  - Alice doesn't lose face…

The Dating Problem with Cards

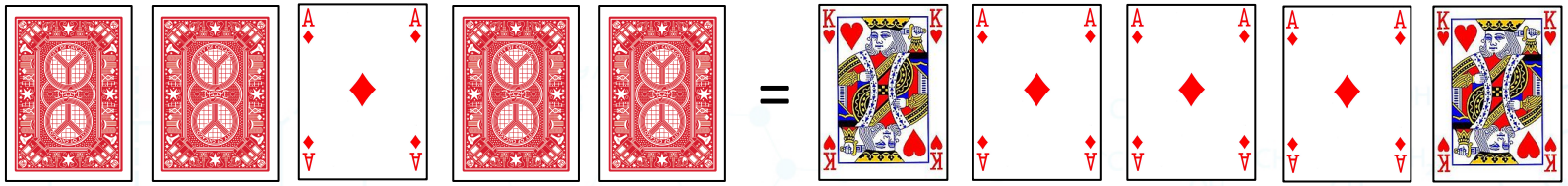
- **Alice and Bob each get two cards** 
- **If Alice likes Bob:**  **and if not:** 
- **If Bob likes Alice:**  **and if not:** 
- **Each turns their cards over, with an Ace in the middle**



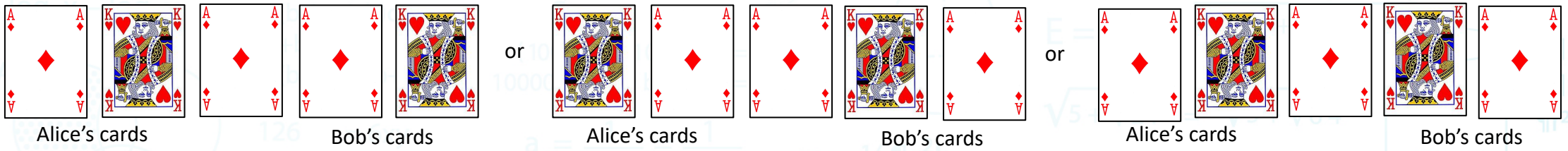
Alice's cards Bob's cards

The Dating Problem with Cards

- **If Alice and Bob like each other**



- **Otherwise,**



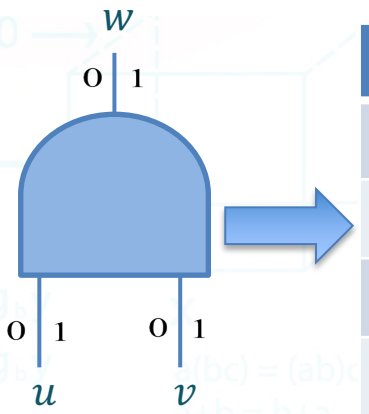
- **Parties turn over middle card and randomly rotate**
- **If three Aces in a row then YES; else NO**

General Secure Computation

- **Powerful feasibility theorems for MPC**
  - Any function can be securely computed!
- **How is it possible to securely compute any function?**
  - Represent the function as a (Boolean or arithmetic) circuit
  - Show how to compute any circuit in MPC
- **Is this even remotely efficient???**

Yao's Garbled Circuits

- **Garbling a single Boolean gate**



u	v	$w = u \wedge v$
0	0	0
0	1	0
1	0	0
1	1	1

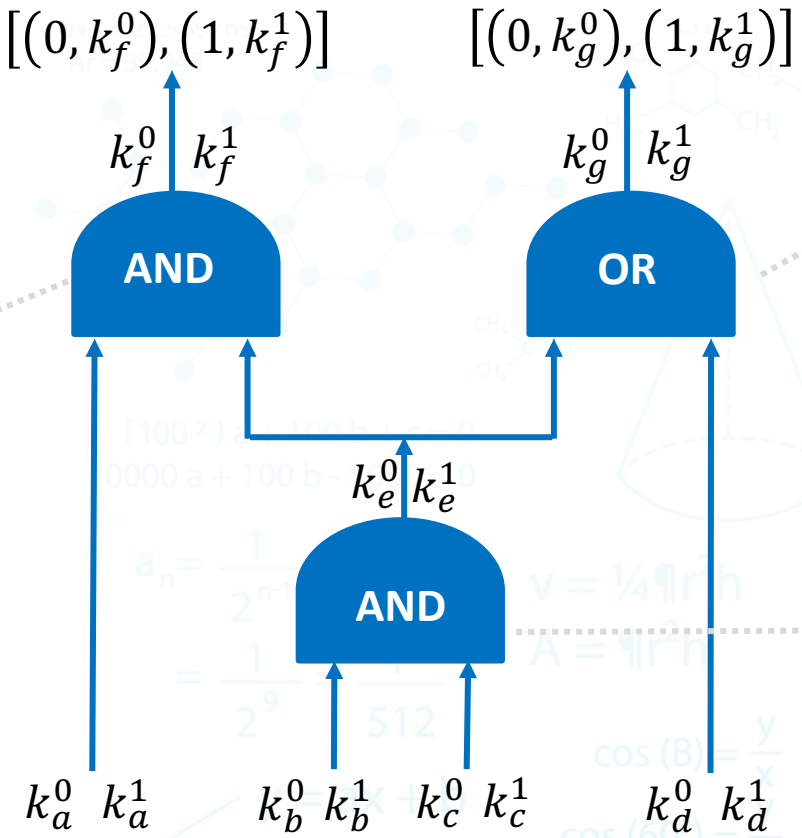
u	v	$w = u \wedge v$
k_u^0	k_v^0	k_w^0
k_u^0	k_v^1	k_w^0
k_u^1	k_v^0	k_w^0
k_u^1	k_v^1	k_w^1

$$E_{k_u^0}\left(E_{k_v^0}(k_w^0)\right)$$
$$E_{k_u^0}\left(E_{k_v^1}(k_w^0)\right)$$
$$E_{k_u^1}\left(E_{k_v^0}(k_w^0)\right)$$
$$E_{k_u^1}\left(E_{k_v^1}(k_w^1)\right)$$

In random order

- Given one key on each input wire, can compute the key on the output wire, without learning **anything** about the represented values
- Keys on input wires are called ***garbled inputs***

Garbling an Entire Circuit

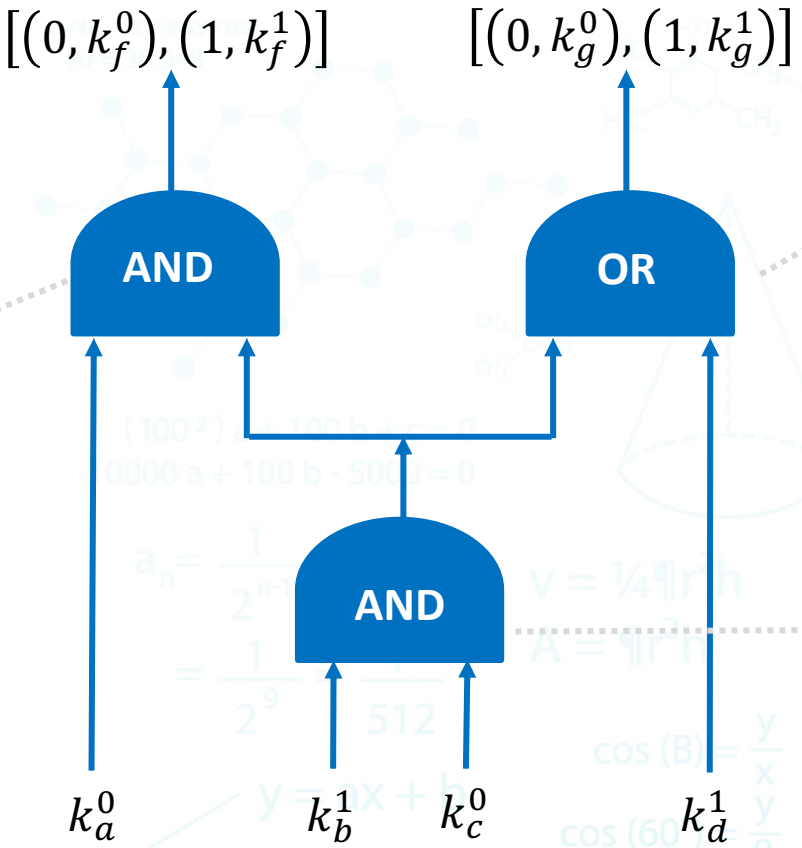


$$E_{k_a^0}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}\left(k_f^1\right)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}\left(k_g^0\right)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}\left(k_g^1\right)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^1}\left(k_e^1\right)\right)$$

Computing a Garbled Circuit



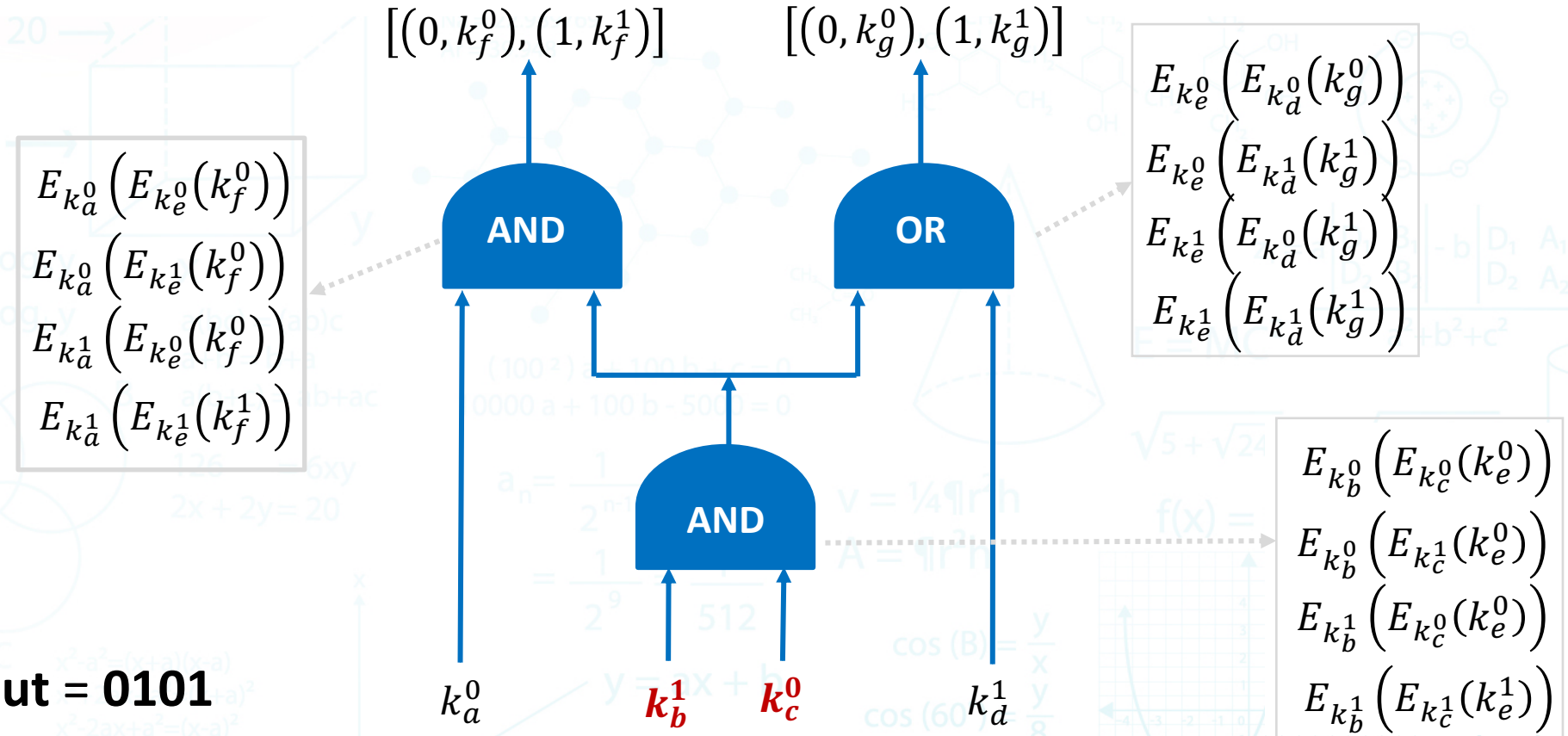
$$E_{k_a^0}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}(k_f^1)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}(k_g^0)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}(k_g^1)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}(k_e^0)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}(k_e^0)\right)$$
$$E_{k_b^1}\left(E_{k_c^0}(k_e^0)\right)$$
$$E_{k_b^1}\left(E_{k_c^1}(k_e^1)\right)$$

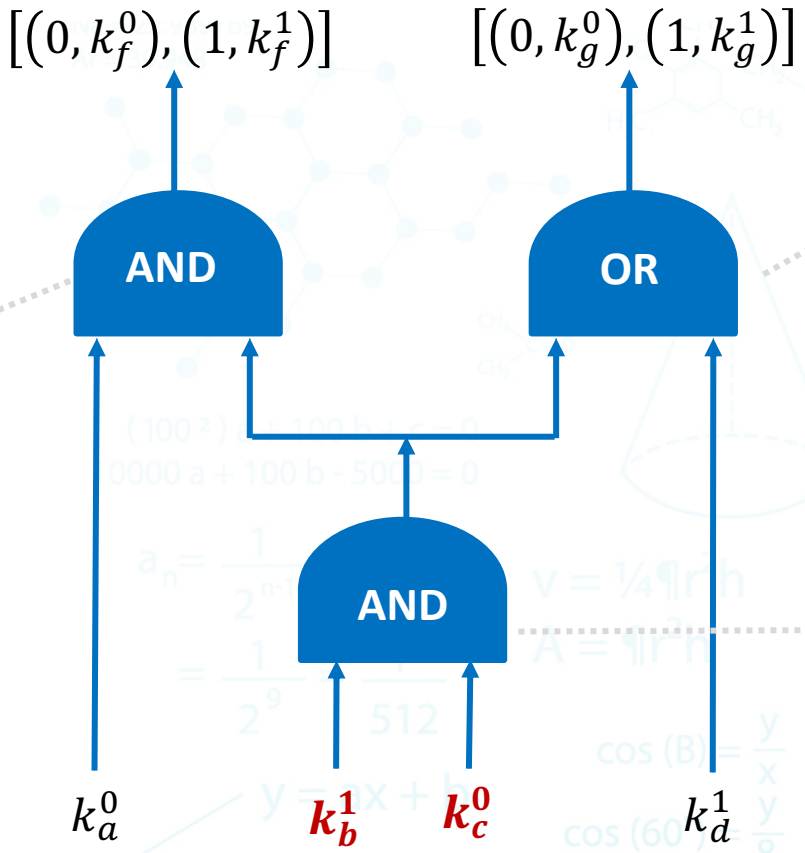
Input = 0101

- **Computing the Garbled Circuit**



Input = 0101

Computing a Garbled Circuit



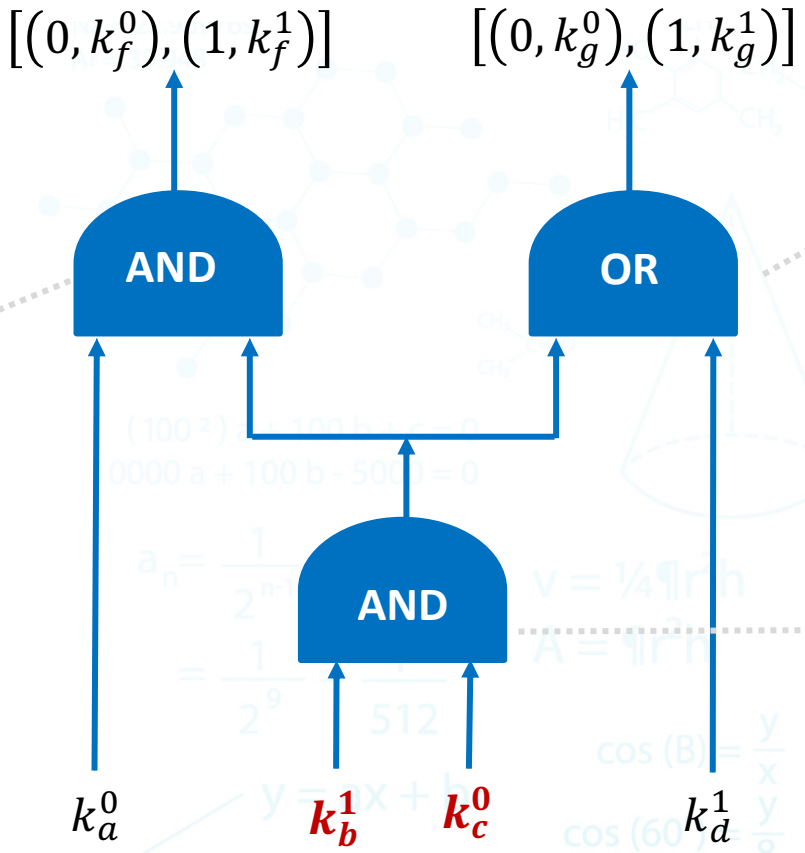
$$E_{k_a^0}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}(k_f^1)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}(k_g^0)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}(k_g^1)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}(k_e^0)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}(k_e^0)\right)$$
$$\boldsymbol{E_{k_b^1}\left(E_{k_c^0}(k_e^0)\right)}$$
$$E_{k_b^1}\left(E_{k_c^1}(k_e^1)\right)$$

Input = 0101

Computing a Garbled Circuit



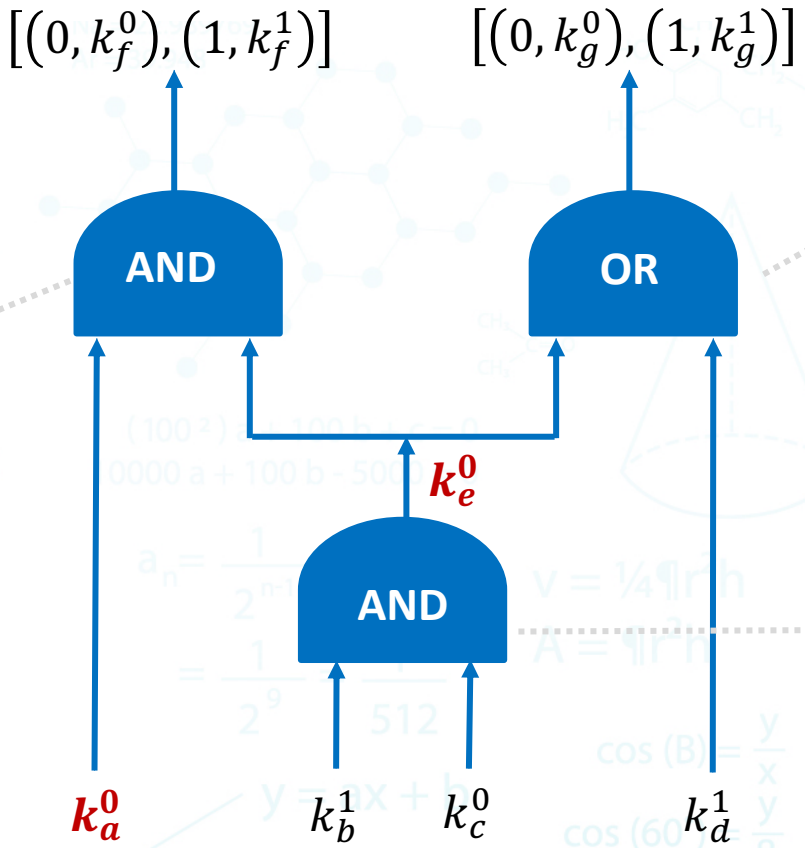
$$E_{k_a^0}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}(k_f^1)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}(k_g^0)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}(k_g^1)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}(k_e^0)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}(k_e^0)\right)$$
$$\boldsymbol{E_{k_b^1}\left(E_{k_c^0}(k_e^0)\right)}$$
$$E_{k_b^1}\left(E_{k_c^1}(k_e^1)\right)$$

Input = 0101

Computing a Garbled Circuit



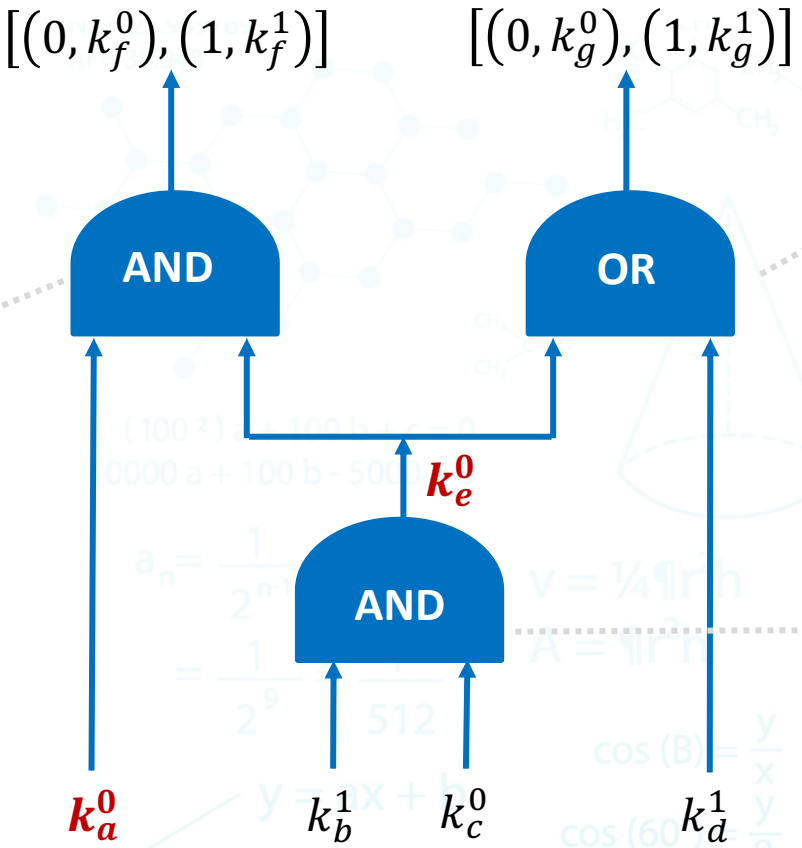
$$E_{k_a^0}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}(k_f^1)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}(k_g^0)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}(k_g^1)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}(k_e^0)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}(k_e^0)\right)$$
$$\boldsymbol{E_{k_b^1}\left(E_{k_c^0}(k_e^0)\right)}$$
$$E_{k_b^1}\left(E_{k_c^1}(k_e^1)\right)$$

Input = 0101

Computing a Garbled Circuit



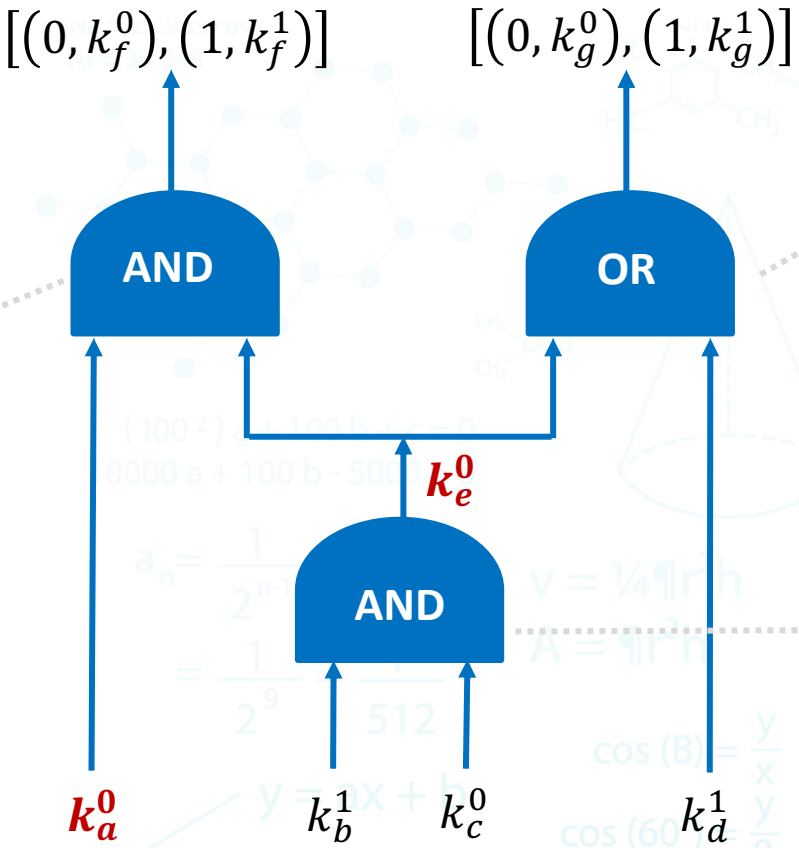
$$E_{k_a^0}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}\left(k_f^1\right)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}\left(k_g^0\right)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}\left(k_g^1\right)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^1}\left(k_e^1\right)\right)$$

Input = 0101

Computing a Garbled Circuit



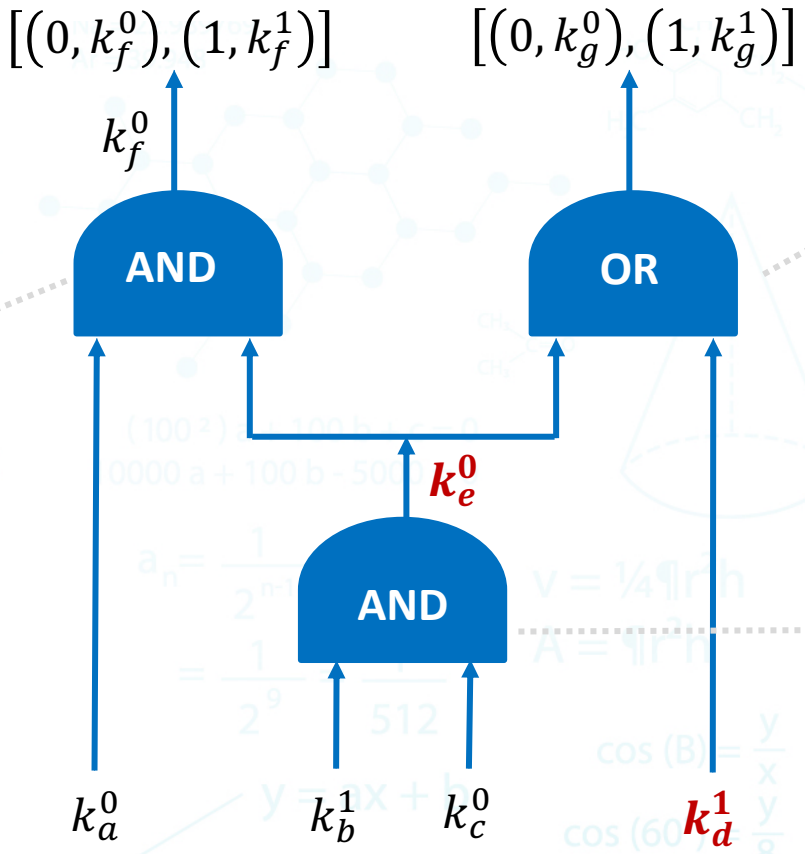
$$E_{k_a^0}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}\left(k_f^1\right)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}\left(k_g^0\right)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}\left(k_g^1\right)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^1}\left(k_e^1\right)\right)$$

Input = 0101

Computing a Garbled Circuit



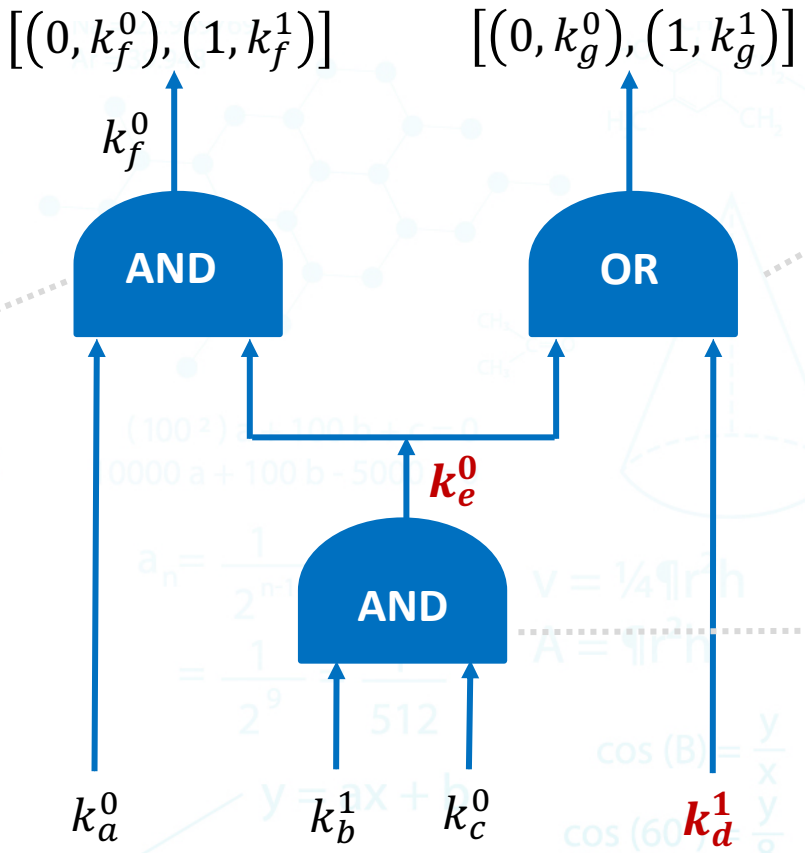
$$E_{k_a^0}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}\left(k_f^1\right)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}\left(k_g^0\right)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}\left(k_g^1\right)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^1}\left(k_e^1\right)\right)$$

Input = 0101

Computing a Garbled Circuit



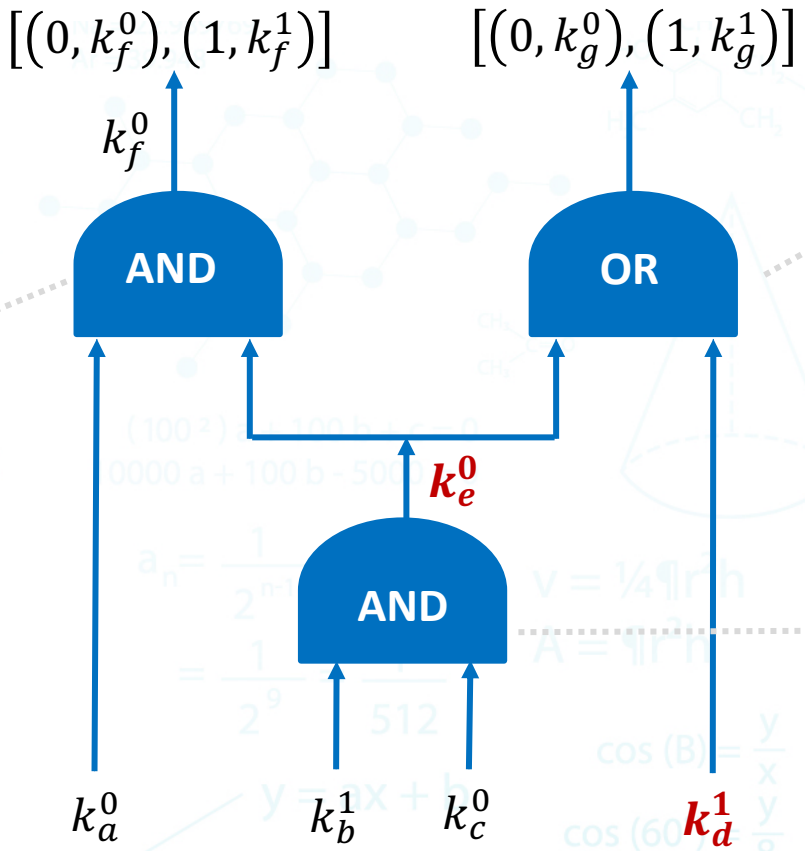
$$E_{k_a^0}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}(k_f^1)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}(k_g^0)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}(k_g^1)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}(k_e^0)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}(k_e^0)\right)$$
$$E_{k_b^1}\left(E_{k_c^0}(k_e^0)\right)$$
$$E_{k_b^1}\left(E_{k_c^1}(k_e^1)\right)$$

Input = 0101

Computing a Garbled Circuit



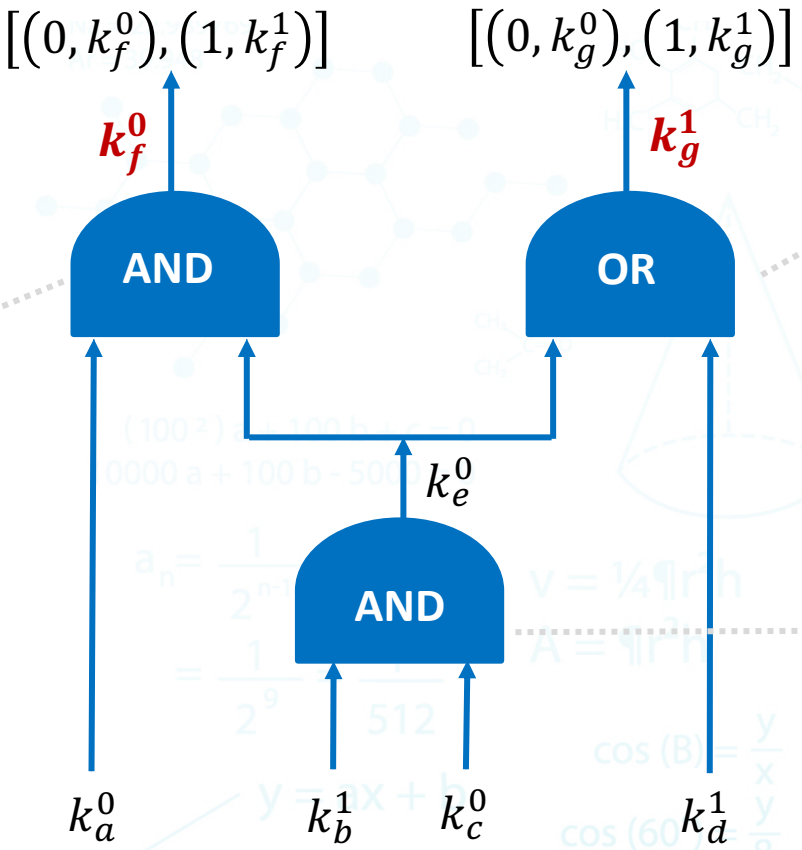
$$E_{k_a^0}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}\left(k_f^1\right)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}\left(k_g^0\right)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}\left(k_g^1\right)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^1}\left(k_e^1\right)\right)$$

Input = 0101

Computing a Garbled Circuit



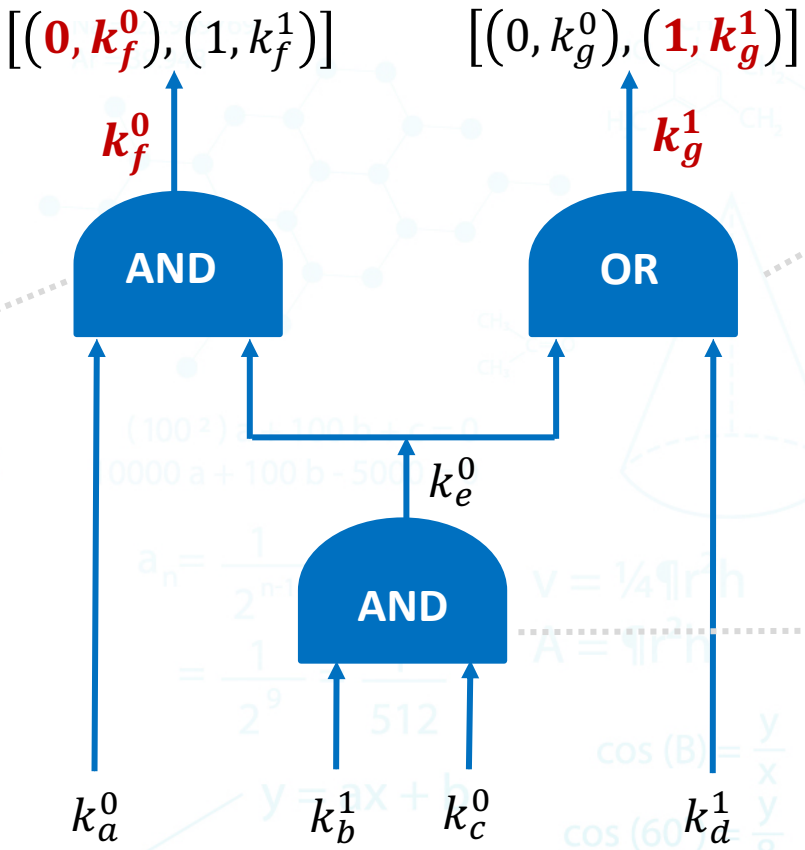
$$E_{k_a^0}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^0}\left(E_{k_e^1}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}\left(k_f^0\right)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}\left(k_f^1\right)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}\left(k_g^0\right)\right)$$
$$E_{k_e^0}\left(E_{k_d^1}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^0}\left(k_g^1\right)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}\left(k_g^1\right)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^0}\left(k_e^0\right)\right)$$
$$E_{k_b^1}\left(E_{k_c^1}\left(k_e^1\right)\right)$$

Input = 0101

Computing a Garbled Circuit



$$\boldsymbol{E_{k_a^0}\left(E_{k_e^0}(k_f^0)\right)}$$
$$E_{k_a^0}\left(E_{k_e^1}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^0}(k_f^0)\right)$$
$$E_{k_a^1}\left(E_{k_e^1}(k_f^1)\right)$$

$$E_{k_e^0}\left(E_{k_d^0}(k_g^0)\right)$$
$$\boldsymbol{E_{k_e^0}\left(E_{k_d^1}(k_g^1)\right)}$$
$$E_{k_e^1}\left(E_{k_d^0}(k_g^1)\right)$$
$$E_{k_e^1}\left(E_{k_d^1}(k_g^1)\right)$$

$$E_{k_b^0}\left(E_{k_c^0}(k_e^0)\right)$$
$$E_{k_b^0}\left(E_{k_c^1}(k_e^0)\right)$$
$$\boldsymbol{E_{k_b^1}\left(E_{k_c^0}(k_e^0)\right)}$$
$$E_{k_b^1}\left(E_{k_c^1}(k_e^1)\right)$$

Input = 0101

Three-Party Protocol with Garbled Circuits

- **Inputs**: party P_1 has input x, and party P_2 has input y; party P_3 has no input
- Protocol with security against one malicious party

- Generate GC from seed
- Generate garbled input of x to garbled circuit

- Generate GC from seed
- Generate garbled input of y to garbled circuit

GC + garbled input of x

GC + garbled input of y

$f(x, y)$

- **Verify** that both garbled circuits are same
- Compute GC on keys and **get output**
- **Send output back** (authenticated)

There exist protocols for two parties with security against one malicious that use garbled circuits (and also multiparty with dishonest majority)

MPC for Specific Tasks

Threshold Cryptography

- **Compute a cryptographic function without any single party holding the key**

- **Motivation:**
  - Make it hard to steal the key
  - Provide quorum authorizations (like signees for bank transactions)

Securely Computing the RSA Function

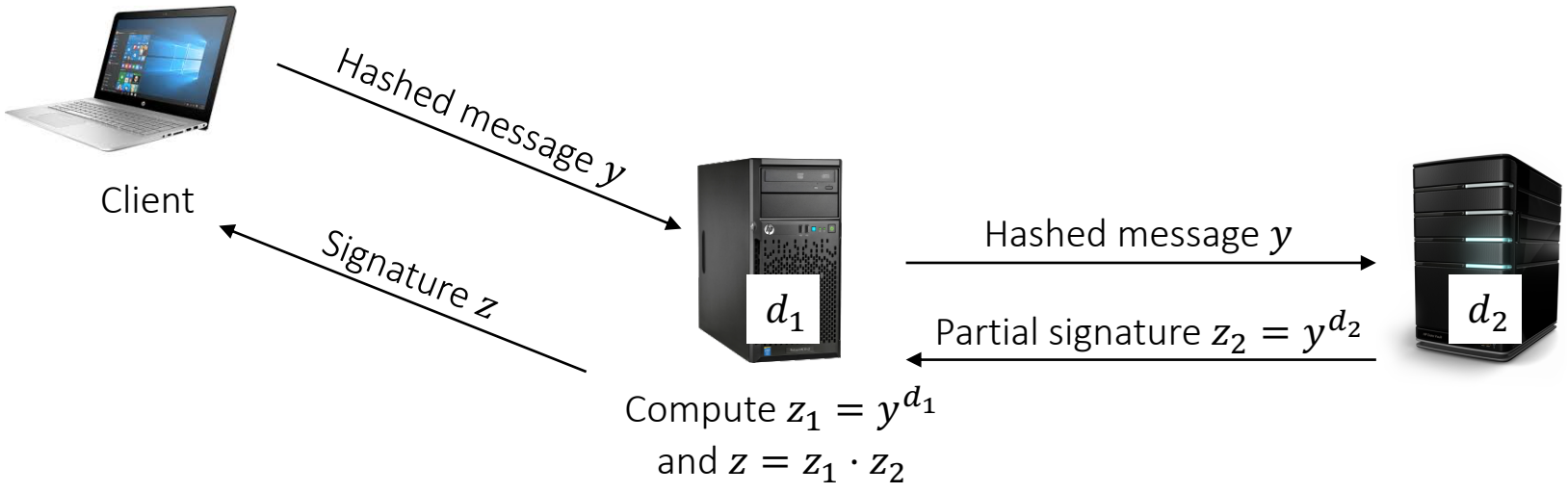
- **RSA signing and decryption:**
  - Private key: (d, N)
  - Public key: (e, N)
  - Private operation (sign/decrypt): $z = y^d \bmod N$
- **RSA key sharing**
  - Server S_1 has a random d_1
  - Server S_2 has $d_2 = d - d_1 \bmod \phi(N)$
  - Note that $d_1 + d_2 = d \bmod \phi(N)$
  - Security:
    - d_1 reveals nothing about d since it's random
    - d_2 reveals nothing about d since d_1 completely hides d

Securely Computing the RSA Function

- **Recall: server S_1 has d_1 and server S_2 has d_2, such that $d_1 + d_2 = d$**
- **Securely computing the private operation $x = y^d \bmod N$**
  - Server S_2 computes $z_2 = y^{d_2} \bmod N$ and sends to server S_1
  - Server S_1 computes $z_1 = y^{d_1} \bmod N$
  - Server S_1 computes $z = z_1 \cdot z_2 \bmod N$
  - Server S_1 *verifies the result* by checking that $y = z^e \bmod N$
  - Note: $z = z_1 \cdot z_2 = y^{d_1} \cdot y^{d_2} = y^{d_1+d_2} = y^d \bmod N$
    - The last equality holds since addition in exponent is mod $\phi(N)$

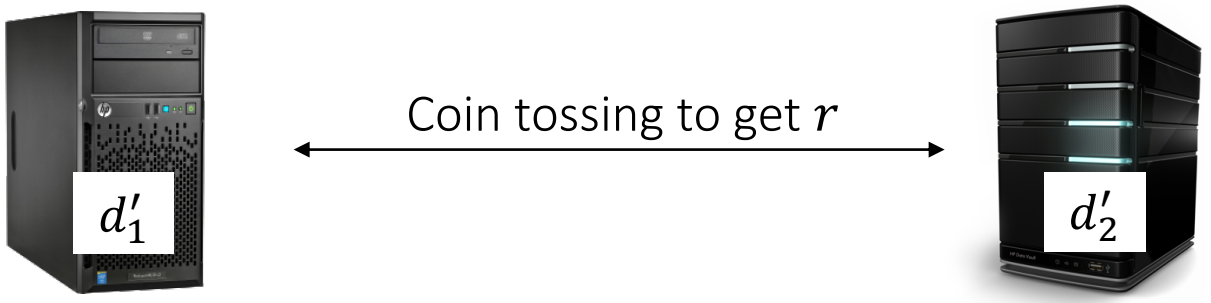
Two-Party Protocol for RSA Decrypt/Sign

- **Inputs: Server S_1 has key share d_1 and hash of message y, and server S_2 has key share d_2**



Secret Share Refresh – Proactive Security

- **At fixed intervals (e.g., every hour), sharing of secret is refreshed**
- **For RSA:**



Compute $d_1' = d_1 + r$

Compute $d_2' = d_2 - r$

- **Note that given d_1 and $d_2' = d_2 - r$, nothing can be learned about d**

Other Threshold Cryptography

- **As with RSA, it is possible to efficiently compute ECDH, ECDSA, etc.**
  - They all have (mostly) nice algebraic structure
- **What about AES, HMAC, and so on?**
  - As above, convert the function description to a Boolean circuit (AND/XOR gates)
  - Use garbled circuits, or another method
- **Efficiency**
  - AES circuit has about 31,000 gates: 6400 AND and 25000 XOR (but XOR is free)
  - It takes about ½ ms to garble and evaluate an AES circuit
  - We do about 500 AES-256-GCM operations on 32-byte input (key wrap) per second
    - 4-core machine, 10Gbps network

Private Set Intersection

- **The problem:**
  - Input: Alice has a set A, Bob has a set B
  - Output: the set $A \cap B$
- **This problem has many solutions; we will see a conceptually simple one here (for semi-honest adversaries)**
- **Tool – oblivious pseudorandom function evaluation**
  - Input: Alice has a key K, Bob has an input b
  - Output: Alice learns nothing, Bob learns $F_K(b)$
  - Concretely, this could be $AES_K(b)$ or a PRF based on elliptic curves

Protocol for Private Set Intersection

Input: $A = a_1, \dots, a_n$

Choose random K

K →

Oblivious AES Evaluation

← b_i

$y_i = AES_K(b_i)$ →

Locally compute all $x_i = AES_K(a_i)$

$x_1, \dots, x_n$ →

For every i, if $y_i = x_j$ for some j, add b_i to the output set C

← C

Output C

Input: $B = b_1, \dots, b_m$

Output C

Use Cases in Practice

Advertising Conversion – Google

- **The problem:**
  - How can we determine the effectiveness of advertisements for BMWs shown on someone's cellphone?
- **The solution:**
  - Compute how many people were shown the ad on their cellphone
  - Compute how many people who were shown the ad that bought a BMW
  - (Normalize by expected percentage purchase if not shown the ad)
- **Privacy concern: this requires Google and BMW sharing their lists**
- **Solution: use private set intersection**
  - In fact, it suffices to compute the cardinality (or the sum of amount spent)

Boston Wage Gap Study

In 2015, women in Boston earned

77 ¢

Data Snapshot:

- 69 employers
- 113,000 FT employees
- $11 Billion in annual wages

In 2016, women in Boston earned

76 ¢

Data Snapshot:

- 114 employers
- 167,000 FT employees
- $14 Billion in annual wages

In 2018, women in Boston earned

70 ¢

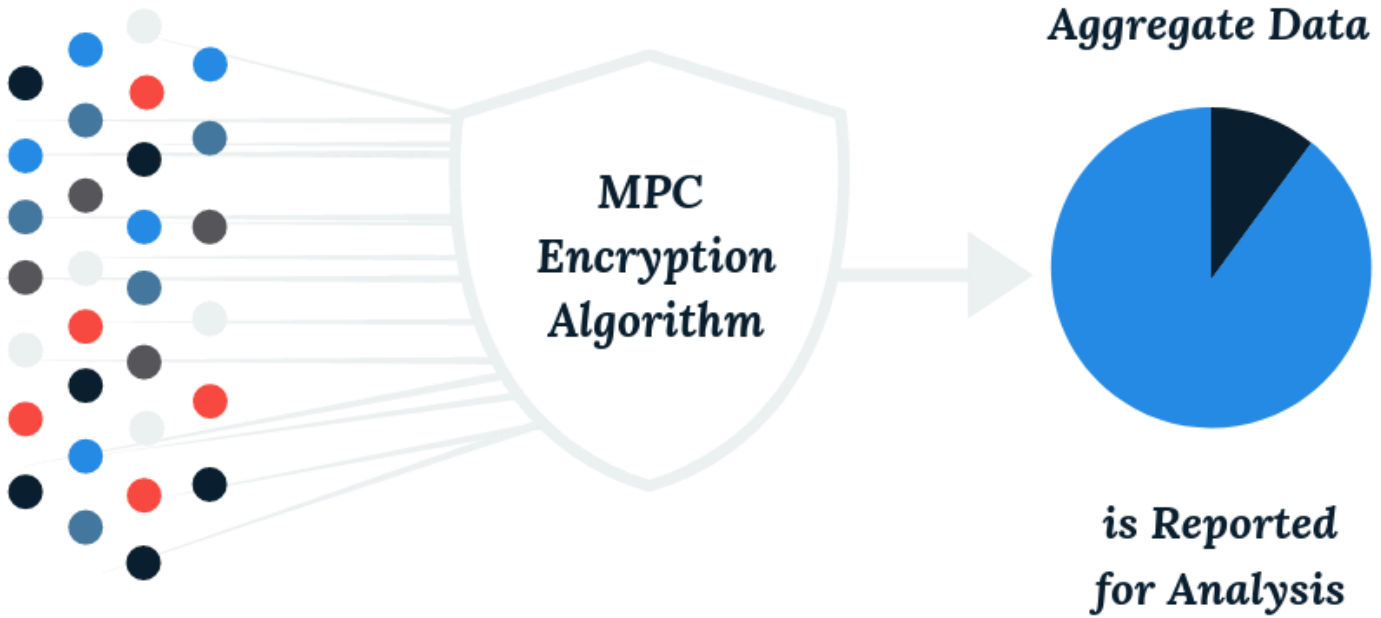
Data Snapshot:

- 125 employers
- 140,000 FT employees
- $12.2 Billion in annual Wages

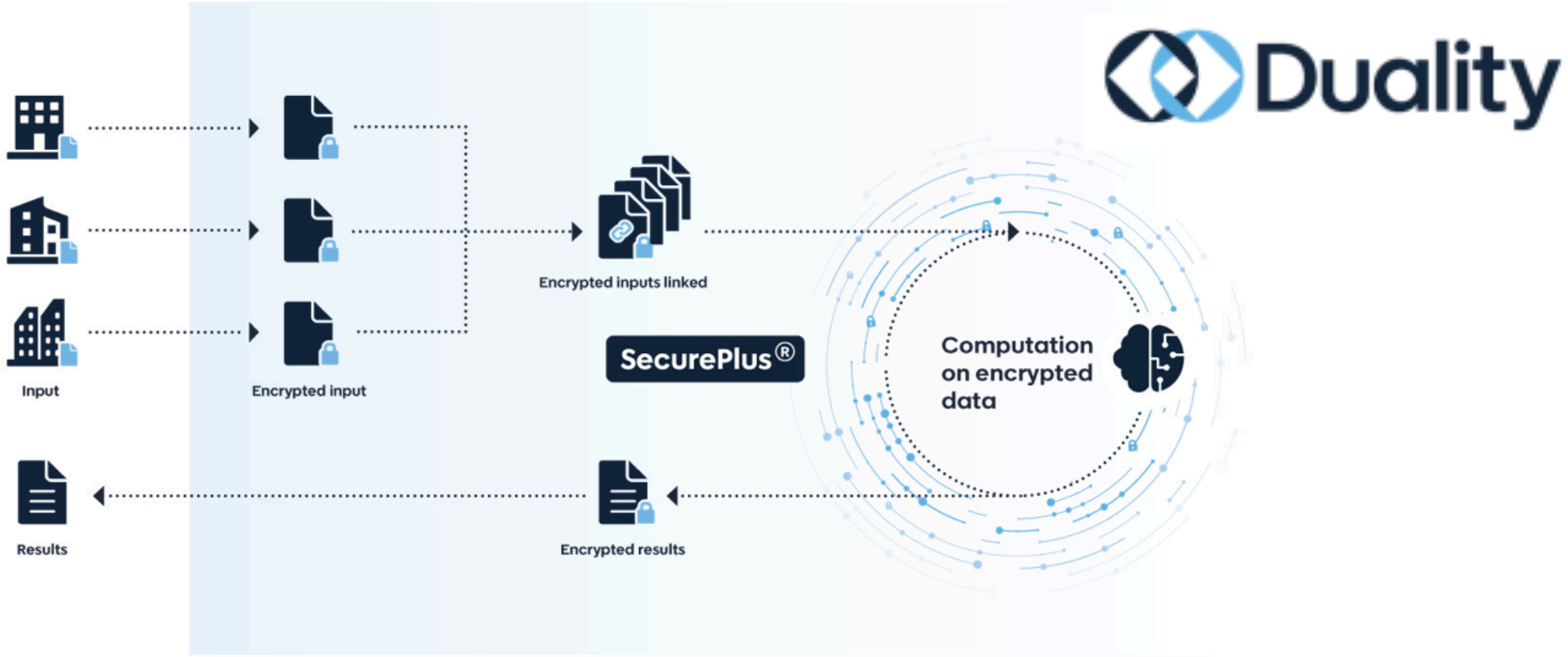
Boston Wage Gap Study

How Does it Work?

Using MPC-backed software, 100% Talent Compact members anonymously provide encrypted data on full-time employees, similar to EEOC-1 job designations. The BWWC then accesses the aggregated data to perform analyses on wage gap by gender, race and ethnicity, and new this year - by industry. ***The more participants, the more secure and accurate the data is.***



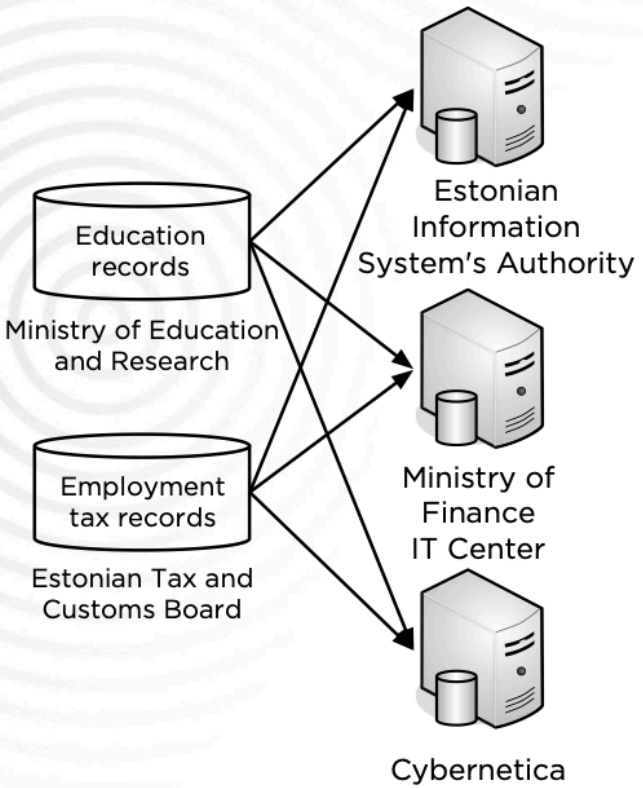
Privacy-Preserving Analytics/Statistics



Enabling privacy-preserving statistical analyses across entities and jurisdictions, Duality SecurePlus Statistics® opens up new opportunities for organizations in regulated industries such as healthcare, financial services, insurance, retail and telecommunications to collaborate with partners on sensitive data so they can grow their business, improve their research, and drive operational efficiencies.

Privacy-Preserving Analytics/Statistics

Sharemind® Powered the Privacy-Preserving Study in the PRIST project



- Source data:
  - 10 million tax records,
  - 600 000 education records.
- Sharemind hosted by government agencies and Cybernetica.
- Data owners used the Sharemind encryption tools to upload data.
- Data never existed outside the source in an unencrypted state.

MPC for Cryptographic Key Protection

- **Classical MPC use cases consider different parties collaborating**
  - Unbound considers where it all belongs to you, but you don't trust your network

1

Each private key is split into random shares stored on separate locations and continually refreshed

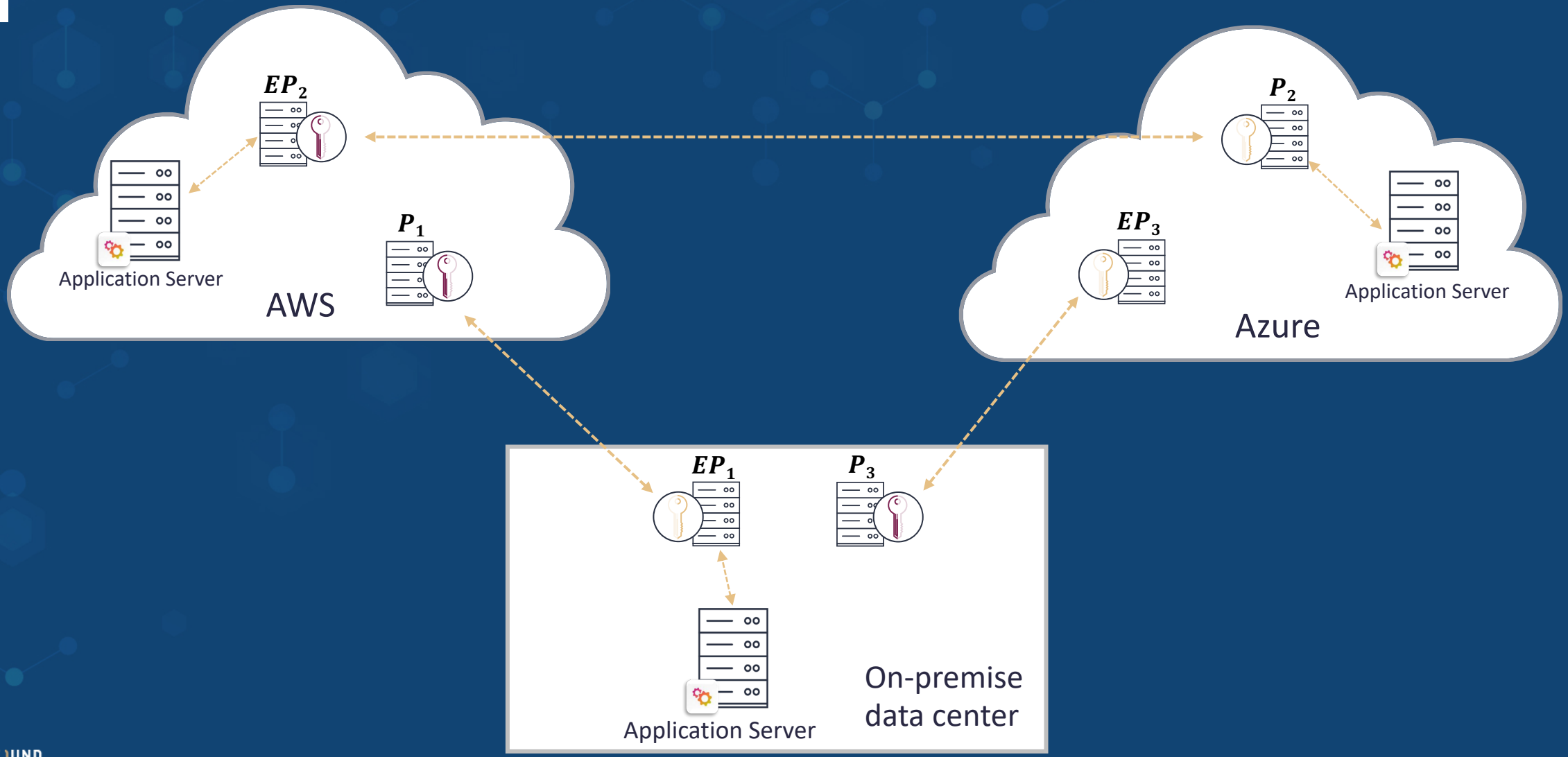
2

Key shares are never united, from generation through usage

3

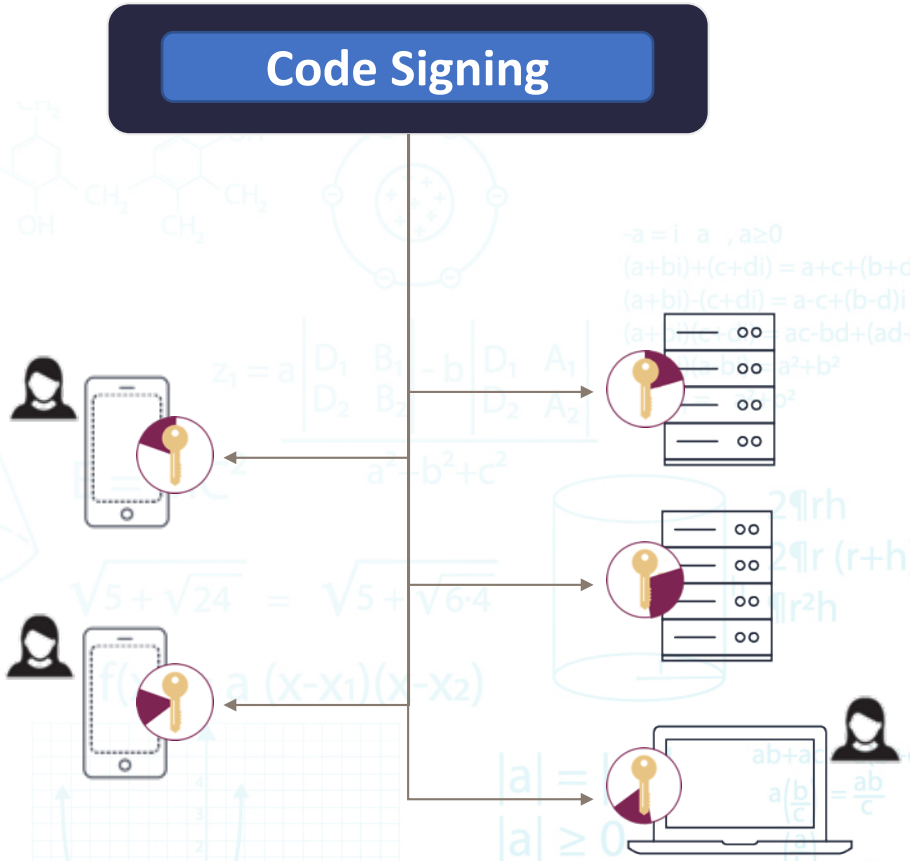
Key material never exists in its entirety at any point of its lifecycle

Deploying an MPC-Based Virtual HSM



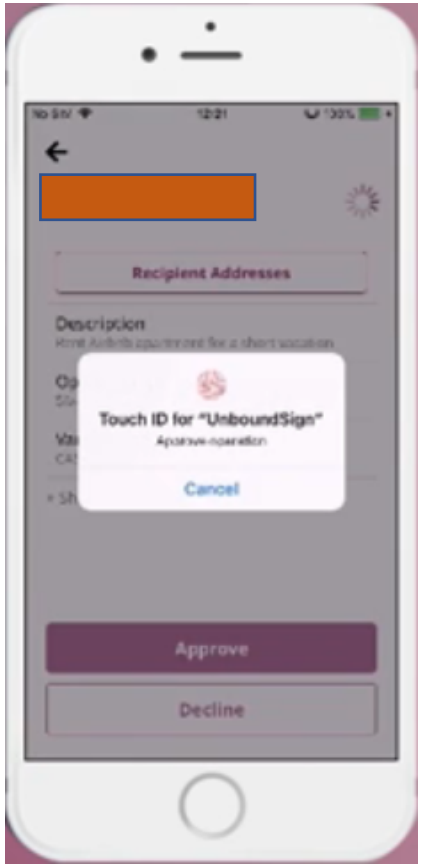
From Key Theft to Key Misuse

- **Consider a code signing application**
  - A single malicious signing is a complete failure
  - Protection from key theft is not enough
- **Using MPC can define flexible quorums based on multiple sets and arbitrary thresholds**
  - 2 out of the 3 parties at R&D, AND
  - 1 out of the 2 parties at legal
- **Can set quorum sizes depending on need**
- **All parties participate and so approval is cryptographically enforced**



Two-Factor Authentication with MPC

- **Mobiles are powerful computing devices, but extremely vulnerable**
- **Virtual smartcard / OTP token on mobile**
  - Mobile and server hold key shares and compute via MPC
    - Key never present on mobile at any time
  - Refresh key sharing at *every single operation*
    - Strong anti-cloning and detection
  - All operations are audited at the server as well as mobile
    - Full visibility into operations
  - Easy to use – mobile is always with you
    - This is a big security advantage
  - Easy deployment and management



Summary

- **MPC is a mature technology and ready for deployment**
- **MPC still requires high expertise to deploy**
  - What problems can be solved efficiently?
  - Tailoring protocols to specific needs
  - Subtleties in published protocols (papers almost never specific basic checks)
- **MPC is being used in production, and its use and interest are quickly growing**


UNBOUND
WHERE SECURITY IS KEY