Medical Data Privacy and Ethics in the Age of Artificial Intelligence

Lecture 16: Cryptographic Privacy-Preserving Computing in Medicine

Zhiyu Wan, PhD (wanzhy@shanghaitech.edu.cn)

Assistant Professor of Biomedical Engineering

ShanghaiTech University

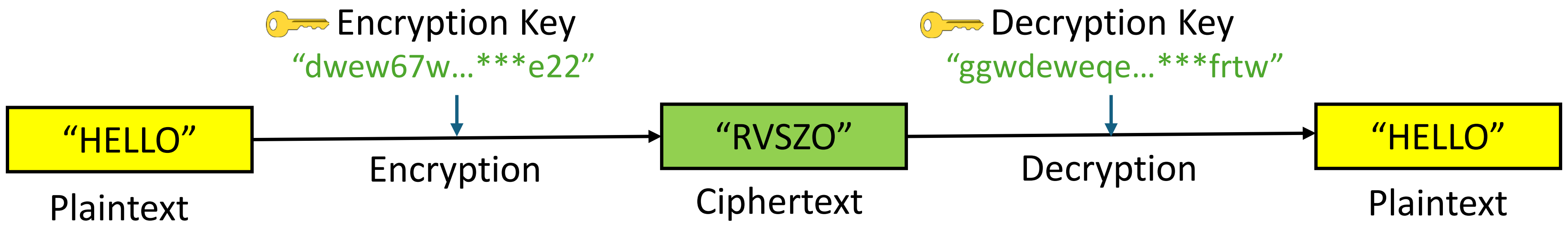
May 9, 2025

Learning Objectives of This Lecture

After this lecture, students should be able to:

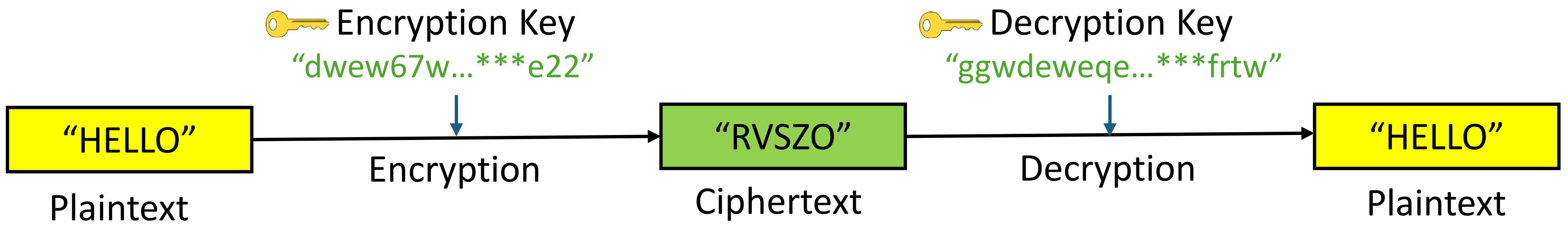
- Know the basics of cryptography
  - Symmetric and asymmetric cryptography
- Know the concept of homomorphic encryption (HE)
  - Partially HE and Fully HE
- Know the concept of secure multi-party computation (SMPC)
  - Secret Sharing
  - Garbled Circuits
  - Oblivious Transfer
  - Zero-Knowledge Proofs

Cryptography



- Cryptography is the science of secure communication and is used to protect information from unauthorized access of modification. It involves the use of mathematical algorithms to encrypt and decrypt data.
  - **Encryption**: The process of converting plaintext (original data) into ciphertext (encoded data) using a cryptographic algorithm and a key.
  - **Decryption**: The reverse process of converting ciphertext back to plaintext using the decryption algorithm and a key.

Cryptography

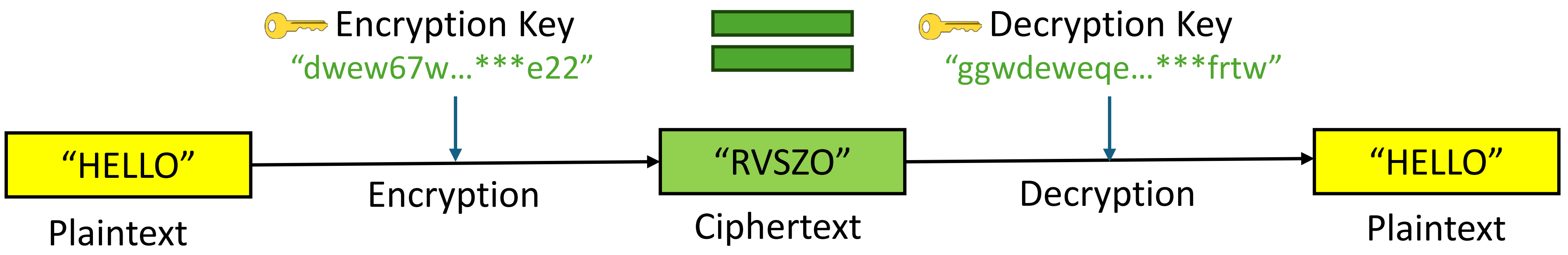


- **Key**: A key is a piece of information used by the cryptographic algorithm to perform encryption or decryption.
- **Algorithm**: A mathematical procedure or set of rules used to perform encryption or decryption.
- **Plaintext**: Plaintext refers to the original, human-readable data or information before any encryption or transformation.
- **Ciphertext**: Ciphertext is the result of applying an encryption algorithm to the plaintext using a key. It is the encrypted, unreadable form of the original data.

Goals in Cryptography

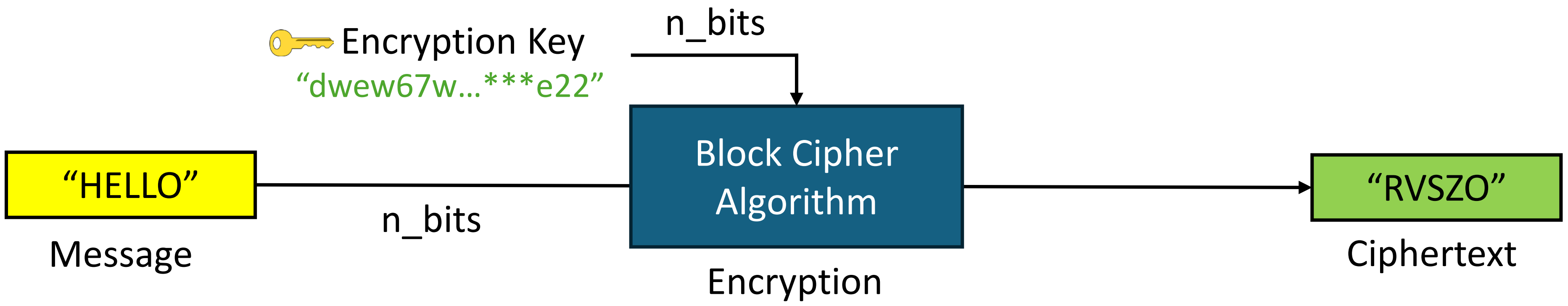
- **Confidentiality**: Ensuring that unauthorized parties cannot understand the encrypted data.
- **Integrity**: Verifying that the data has not been altered during transmission or storage.
- **Authentication**: Verifying the identity of the sender or receiver.
- **Non-repudiation**: Preventing a party from denying the authenticity of their signature or message.

Symmetric Cryptography



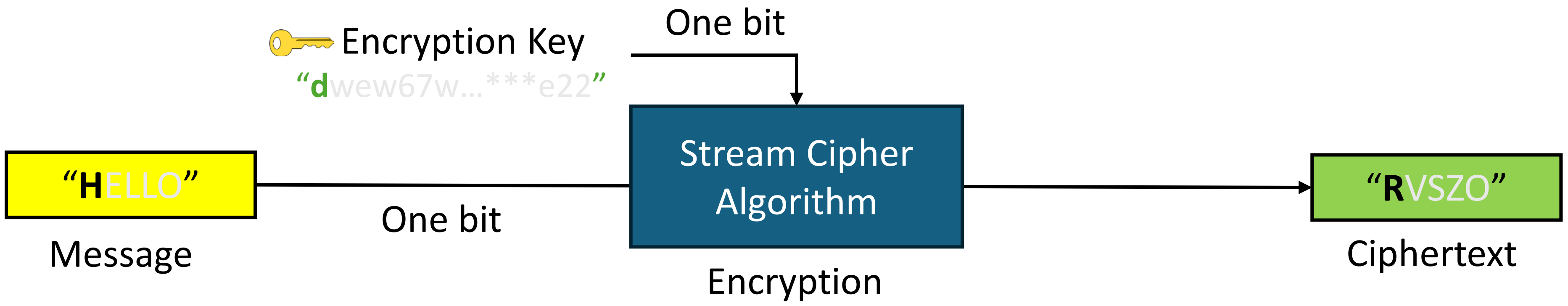
- Only one key for encryption and decryption.
- Also known as secret-key or private-key cryptography.
- Involves key sharing between the involved parties.

Block Ciphers



- Operate on fixed-size blocks of data, dividing the data into blocks and encrypting each block individually.
- The same key is applied to each block independently.
- Are suitable for scenarios where data is processed in fixed-size blocks, such as disk encryption, database encryption, and secure communication protocols.
- Eg. Advanced Encryption Standard (AES), Data Encryption Standard (DES)

Stream Ciphers

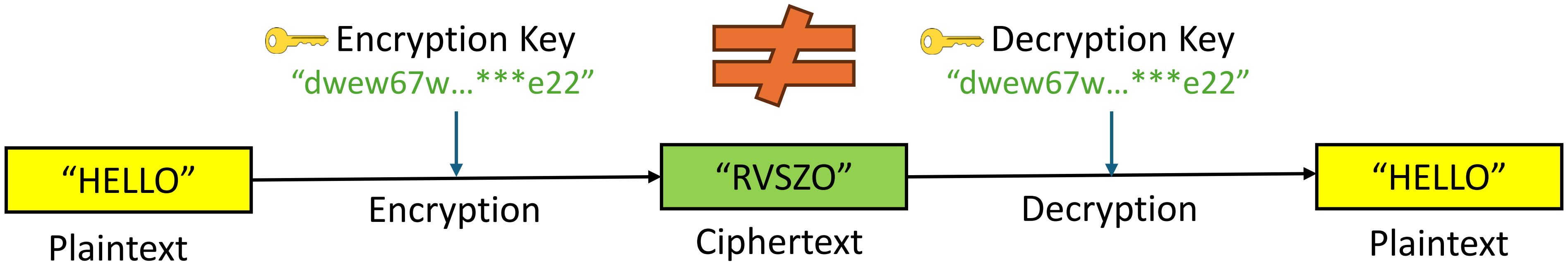


- Encrypt data one bit or byte at a time, generating a continuous stream of key-dependent pseudorandom bits.
- Stream ciphers are suitable for scenarios where data is transmitted continuously, such as real-time communication, wireless communication, and secure streaming.
- Eg. RC4, ChaCha20.

Pros and Cons of Symmetric Cryptography

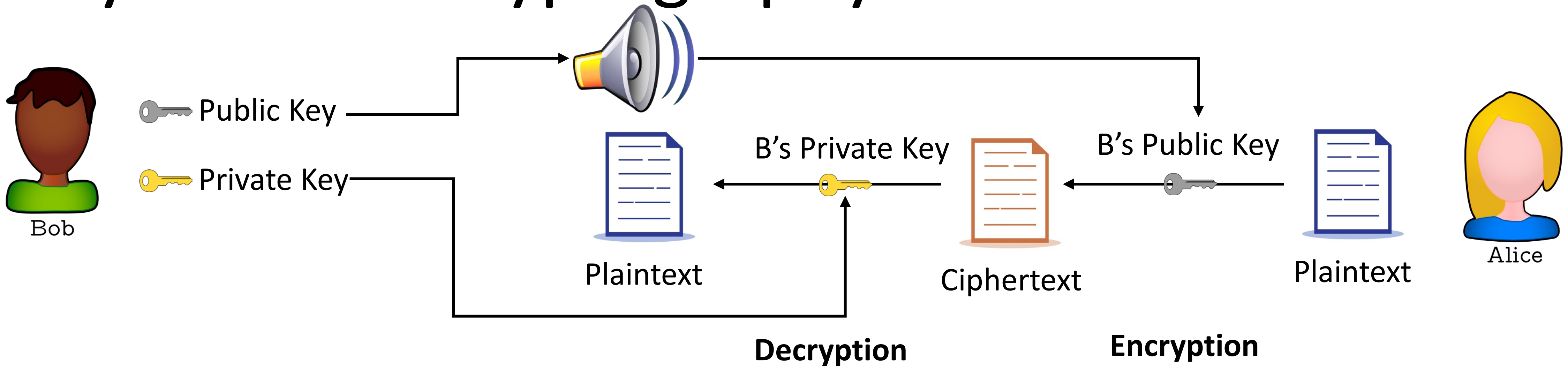
- Advantages
  - **Speed**: Symmetric algorithms are generally faster than asymmetric algorithms.
  - **Simplicity**: Implementation and management of symmetric key systems are often simpler.
- Disadvantages:
  - **Key Distribution**: Securely sharing and managing the key between parties can be challenging.
  - **Lack of Forward Secrecy**: If the key is compromised, all past and future communication is at risk.

Asymmetric Cryptography



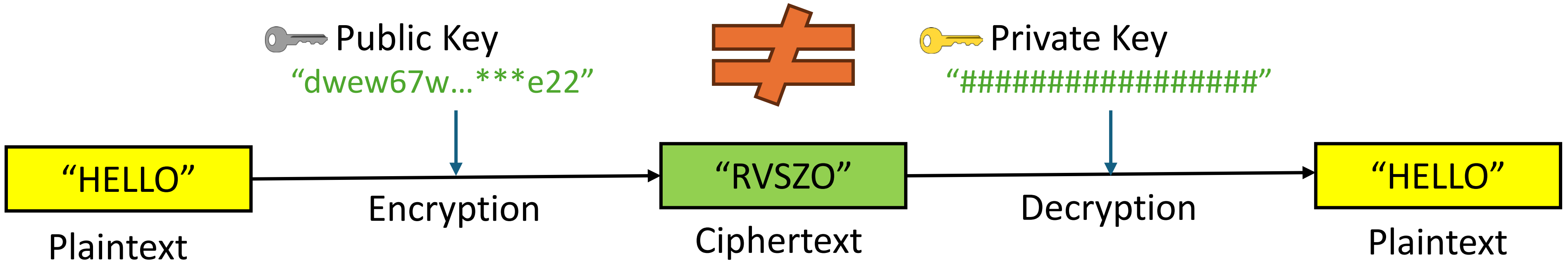
- Two different keys for encryption and decryption.
- Also known as public-key cryptography.
- Two types of keys:
  - **Public key**: Shared with everyone.
  - **Private key**: kept secret.

Asymmetric Cryptography



- If User A wants to send an encrypted message to User B:
  1. User B creates a key pair of a Public and a Private key.
  2. User B broadcasts the public key.
  3. User A uses User B's public key to encrypt the message.
  4. User B, the recipient, decrypts the message using his own private key.

Asymmetric Cryptography

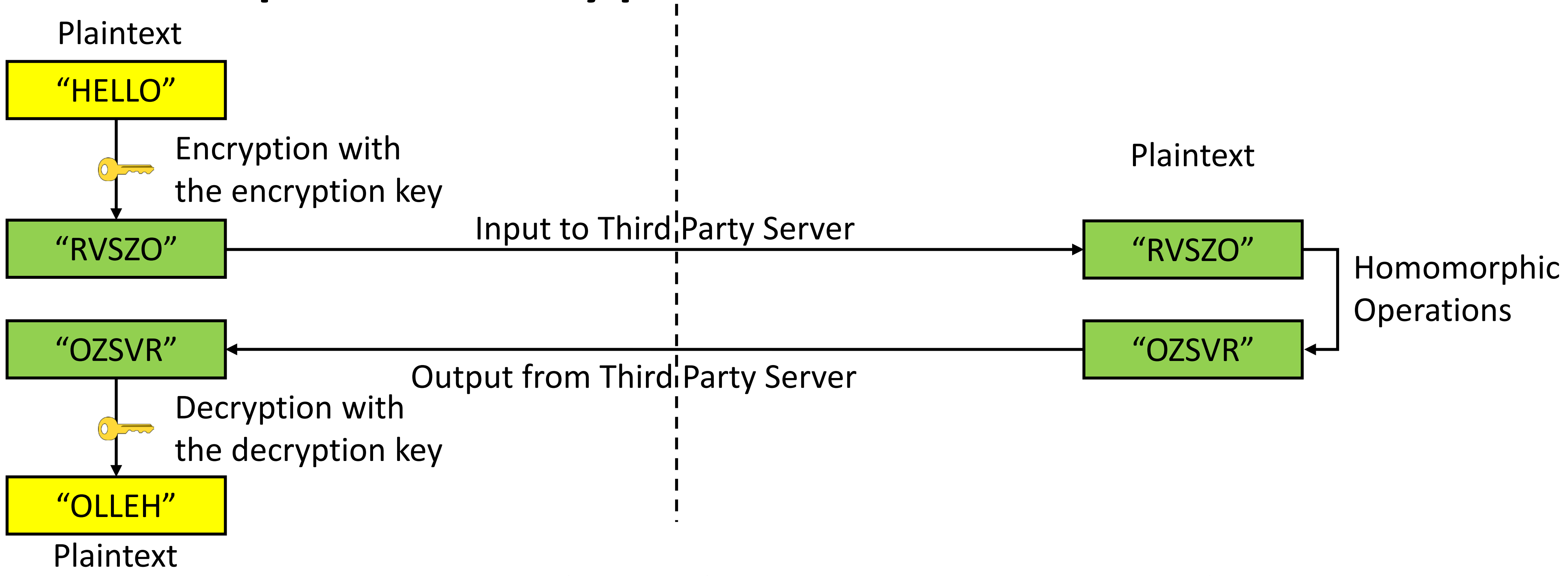


- Asymmetric encryption is often used for creating digital signatures. A user can sign a message with their private key, and others can verify authenticity of the signature using the corresponding public key.
  - **RSA (Rivest-Shamir-Adleman)**: Widely used for secure data transmission and digital signatures.
  - **DSA (Digital Signature Algorithm)**: Primarily used for creating digital signatures.
  - **ECC (Elliptic Curve Cryptography)**: Known for providing strong security with shorter key lengths compared to other asymmetric algorithms.

Pros and Cons of Asymmetric Cryptography

- Advantages
  - **Key Distribution**: No need for secure key exchange; each user has a public-private key pair.
  - **Digital Signatures**: Allows for secure authentication and non-repudiation.
- Disadvantages:
  - **Computational Cost**: Asymmetric algorithms are generally slower than symmetric algorithms.
  - **Key Length**: Longer keys are needed for equivalent security, leading to increased computational overhead.

Homomorphic Encryption



Plaintext
"HELLO"
Encryption with the encryption key
"RVSZO"
Input to Third Party Server
Plaintext
"RVSZO"
Homomorphic Operations
"OZSVR"
Output from Third Party Server
"OZSVR"
Decryption with the decryption key
"OLLEH"
Plaintext

- Homomorphic encryption is a form of encryption that allows for computations to be performed on encrypted data without decrypting it first. This means that computations can be carried out on sensitive data while it remains encrypted, preserving privacy and confidentiality.

Homomorphic Encryption

- Privacy-Preserving
  - Homomorphic encryption (HE) enables computations to be performed on encrypted data without revealing the underlying plaintext.
- Homomorphism
  - The term “homomorphic” refers to a property of the encryption scheme that allows certain algebraic operations to be performed on ciphertexts, resulting in equivalent operations on the plaintexts when decrypted.
- Security
  - HE schemes are designed to be secure against various cryptographic attacks.
  - The security of these schemes relies on underlying mathematical assumptions like the hardness of certain mathematical problems like factoring large integers or computing discrete logarithms.
- Performance
  - HE typically incurs a computational overheard compared to traditional encryption schemes due to the complexity of performing computations on encrypted data.

Types of Homomorphic Encryption

- Partially Homomorphic Encryption (PHE)
  - Supports only one type of algebraic operation (e.g., addition or multiplication) on encrypted data.
- Somewhat Homomorphic Encryption (SHE)
  - Supports a limited number of algebraic operations (e.g. addition and multiplication) on encrypted data.
- Fully Homomorphic Encryption (FHE)
  - Supports arbitrary computations on encrypted data, including addition, multiplication, and more complex operations.

HE Early Development (1970s-2000s)

Rivest RL, Adleman L, Dertouzos ML. **On data banks and privacy homomorphisms**. *Foundations of secure computation*. **1978** Oct 16;4(11):169-80.

- This paper explored the idea of performing computations on encrypted data without decrypting it.

Gentry C. **A fully homomorphic encryption scheme**. *Stanford university*; 2009.

- Craig Gentry proposed Fully Homomorphic in his doctoral thesis.
- He proposed the first practical FHE scheme, which enabled arbitrary computations on encrypted data.
- IBM has been a major player in the development of HE.
- Researchers at IBM, including Craig Gentry, Shai Halevi, & Victor Shoup, have contributed a lot to advancing the theory and practical implementations of HE.

Homomorphic Operations

- Homomorphic Addition
  - If $E(a)$ and $E(b)$ are ciphertexts representing encrypted values of 'a' and 'b', then $E(a+b)$ can be computed directly without decrypting $E(a)$ or $E(b)$.
  - Useful in secure voting system or privacy-preserving aggregation of data.
  - Eg. The Paillier cryptosystem.
- Homomorphic Multiplication
  - If $E(a)$ and $E(b)$ are ciphertexts representing encrypted values of 'a' and 'b', then $E(a \times b)$ can be computed directly without decrypting $E(a)$ or $E(b)$.
  - Useful in secure multiplication of private values in cryptographic protocols.
  - Eg. The Goldwasser-Micali cryptosystem.
- Combining Operations
  - Allows both homomorphic addition and homomorphic multiplication operations to be performed on encrypted data.

Partially Homomorphic Encryption (PHE)

- PHE is a type of HE that supports the evaluation of only one specific type of mathematical operation (e.g., addition or multiplication) on encrypted data.

- Advantages
  - **More Efficient**: PHE has the advantage of being more efficient compared to fully homomorphic encryption.
- Disadvantages
  - **Limited Operations**: Utility is restricted to specific types of operations.
  - **Complexity of Computations**: Complex computations requiring both addition and multiplication on encrypted data cannot be directly performed with PHE.
  - **Security Considerations**: The security of PHE schemes relies on the underlying mathematical assumptions, and users must carefully consider the guarantees.

Fully Homomorphic Encryption (FHE)

- Advantages
  - FHE allows for both addition and multiplication operations to be performed on encrypted data.
  - Supports chaining multiple operations together, enabling complex computations on encrypted data.
  - FHE systems exhibit a high degree of homomorphism, preserving the structure of the operations on the encrypted data.
- Disadvantages
  - FHE causes accumulation of noise during repeated homomorphic operations, which may lead to error in the decrypted results.
  - FHE is computationally intensive, and it incurs significant overhead.

Fully Homomorphic Encryption (FHE)

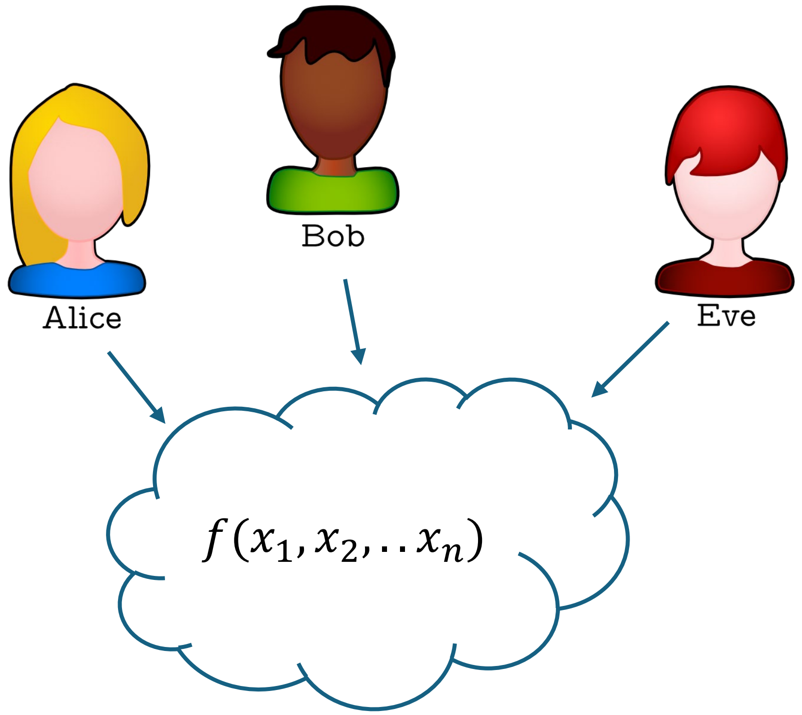
- Applications
  - Secure Outsourcing of Computations.
  - Privacy-Preserving Data Analysis.
  - Secure Multi-Party Computation.
- Schemes and Libraries
  - **Schemes**: Original Gentry’s scheme, the Brakerski-Vaikuntanathan scheme, and others
  - **Libraries**: Microsoft SEAL (Simple Encrypted Arithmetic Library) and TenSEAL.

Homomorphic Encryption Implementation Tools

- Microsoft SEAL (Simple Encrypted Arithmetic Library)
  - It supports both PHE and FHE schemes.
- TenSEAL
  - An open-source library that builds on Microsoft SEAL and extends its capabilities.
  - It is designed to simplify the development of HE applications.
- PySEAL
  - A Python wrapper for Microsoft SEAL
- TF-Encrypted
  - An extension of the TensorFlow framework. It allows for the execution of machine learning models on encrypted inputs.

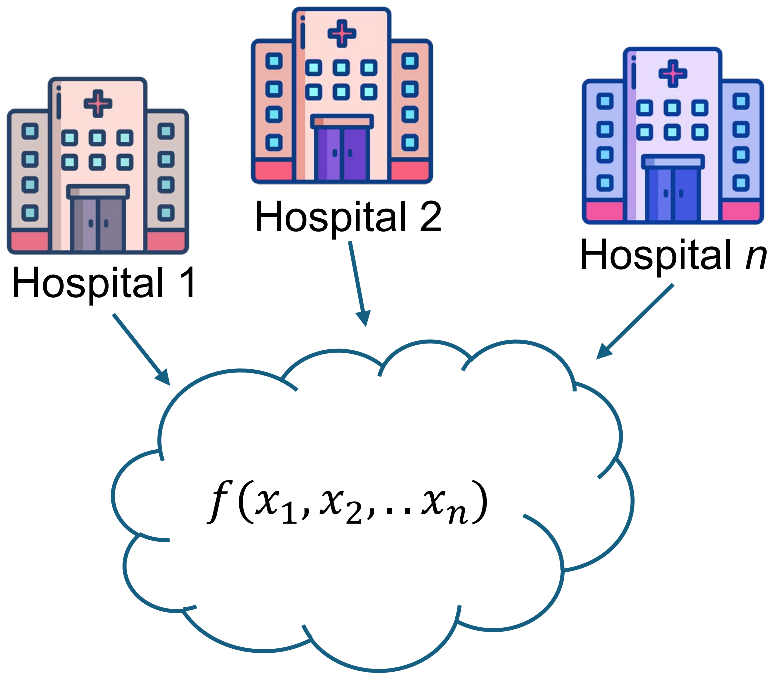
Secure Multi-Party Computation (SMPC)

- Secure Multi-Party Computation (SMPC) is a cryptographic technique that allows multiple parties to jointly compute a function over their inputs while keeping those inputs private.
  - The privacy objective of SMPC is to preserve privacy and confidentiality, even when collaborating parties do not fully trust each other or a central authority.
  - SMPC protocols typically involve sophisticated cryptographic techniques such as secret sharing, homomorphic encryption, oblivious transfer, and secure function evaluation.
  - Applications across various domains, including healthcare, finance, data analytics, voting system, etc.



SMPC – Example Use Case

- Hospitals want to collaborate to diagnose a rare disease but need to protect patient privacy.
  - **Data Preparation**: Each hospital preprocesses its patient data, ensuring that personally identifiable information (PII) is removed or encrypted.
  - **Protocol Design**: A protocol for disease diagnosis is designed using SMPC techniques.
  - **Secure Computation**: Using SMPC protocols, the hospitals jointly compute the diagnostic model over their respective datasets.
  - **Privacy Preservation**: Throughout the computation, each hospital's input data remains encrypted or hidden from the other parties.
  - **Result Sharing**: Once the computation is complete, the diagnosis result is revealed to all parties.



SMPC – Timeline

Yao AC. **Protocols for secure computations**. In *23rd annual symposium on foundations of computer science* (sfcs 1982) 1982 Nov 3 (pp. 160-164). IEEE.

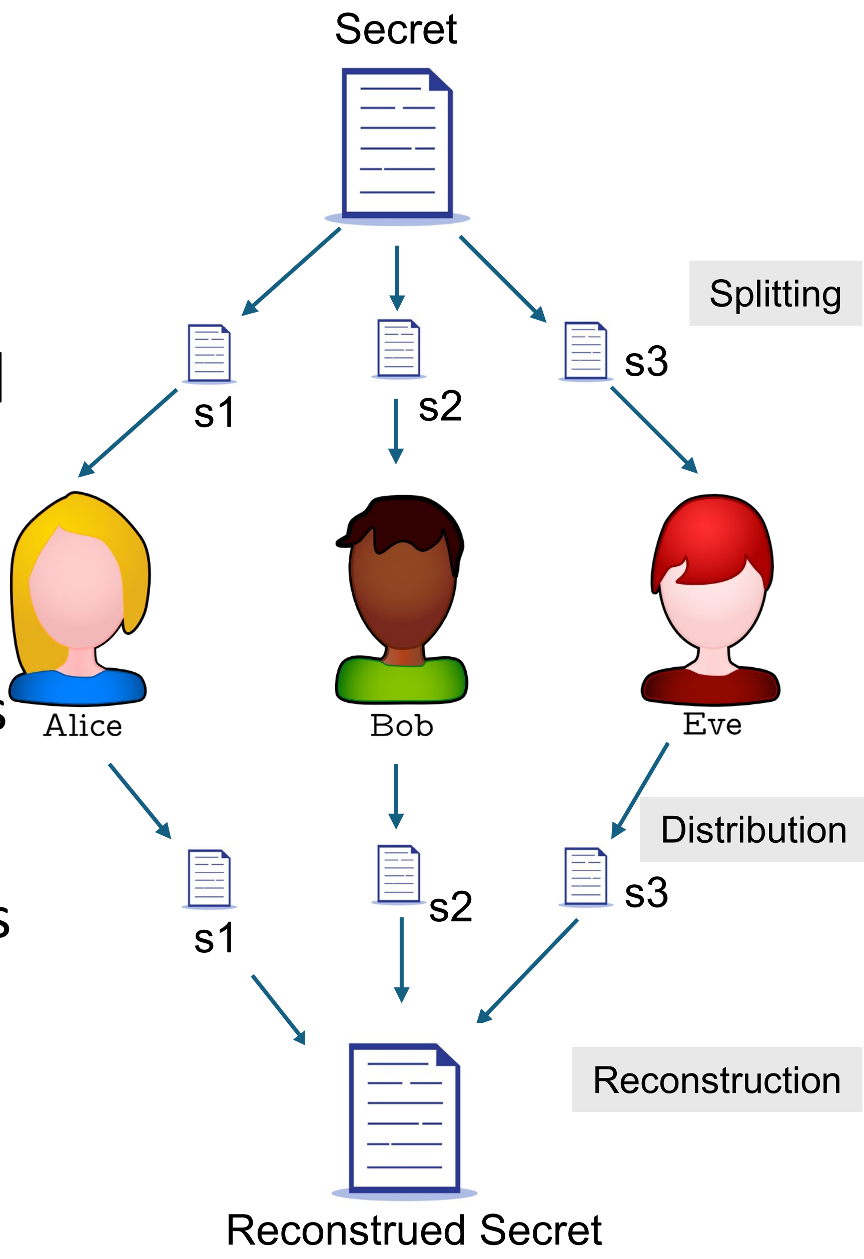
- Andrew Yao’s paper laid the foundation for SMPC.

Goldwasser S, Micali S. **Probabilistic encryption & how to play mental poker keeping secret all partial information**. In *Providing sound foundations for cryptography*: on the work of Shafi Goldwasser and Silvio Micali 2019 Oct 4 (pp. 173-201).

- Later that year Goldwasser and Micali published a high-impact paper.
- Development of Secure Computation Protocols (1980s – 2000s)
- Theoretical Foundations and Complexity Results (1990s – 2000s)
- Practical Implementations and Applications (2000s – Present)

Secret Sharing

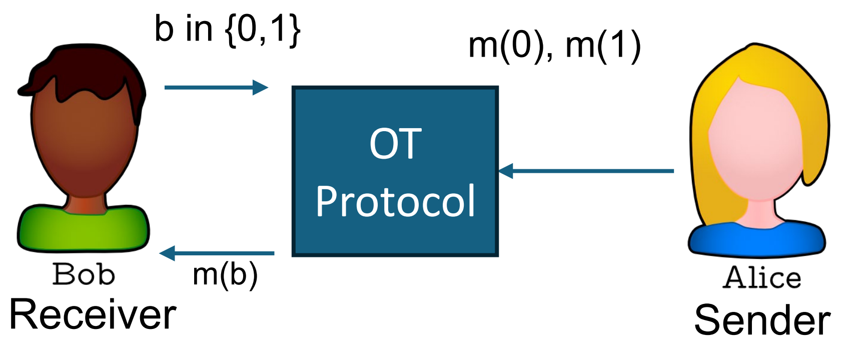
- Secret sharing is a cryptographic technique used to distribute a secret among multiple parties in such a way that no single party can reconstruct the secret on its own.
    - Allows parties to split their private inputs into shares distributed among other parties. Secure computations are performed on these shares, ensuring that no party gains knowledge of the inputs unless a sufficient number of shares are combined.
    - The protocol involves: Secret splitting, distribution, reconstruction.



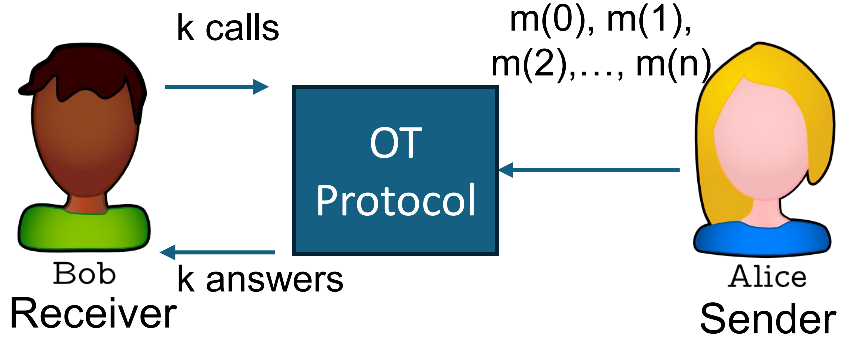
Oblivious Transfer (OT)

- Oblivious Transfer (OT) is a cryptographic protocol that allows a sender to send one of several messages to a receiver without revealing which message was chosen, and the receiver learns only the chosen message.
  - OT protocols enable parties to exchange information in a way that preserves privacy and confidentiality.
  - 1-out-of-2 OT protocol:
    - The sender has two messages (say m0 and m1), and the receiver wants to obtain on of them without the sender knowing which one.
  - k-out-of-n OT protocol:
    - The sender has n messages, and the receiver wants to obtain k of them. The sender learns which messages were requested, but the receiver does not know which messages were not requested.

1-out-of-2 OT



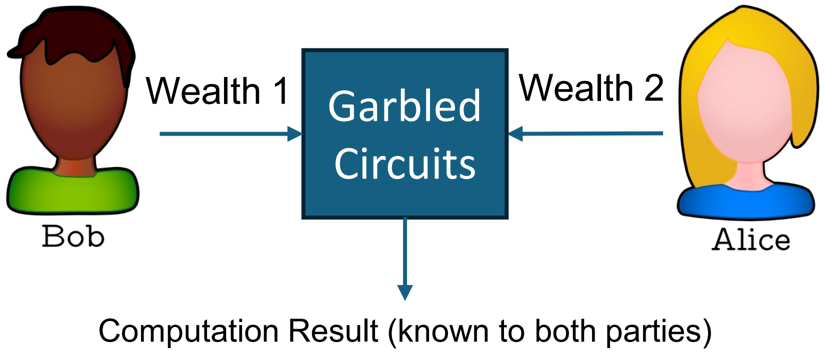
k-out-of-n OT



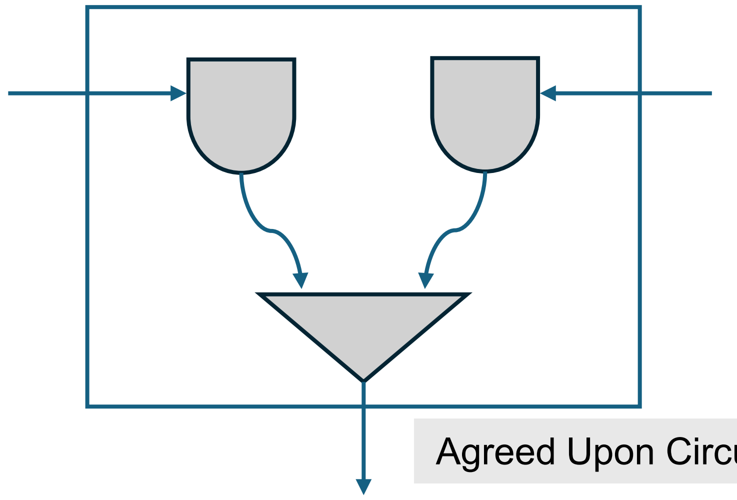
Garbled Circuits

- Garbled circuits are cryptographic constructions used to evaluate functions privately.
  - Each gate in the circuit is garbled such that its inputs and outputs are encoded in a way that hides their values.
  - Enable parties to compute a function on their inputs without revealing those inputs.
  - The protocol involves:
    a. Circuit Description – Parties agree on a Boolean circuit representation of the function they want to compute.
    b. Garbling Phase – Each party garbles its input wire and gates in the input wire and gates in the circuit.
    c. Evaluation Phase – The parties exchange their garbled input wires and evaluate the garbled circuit gate-by-gate.

Yao's Millionaires' Problem

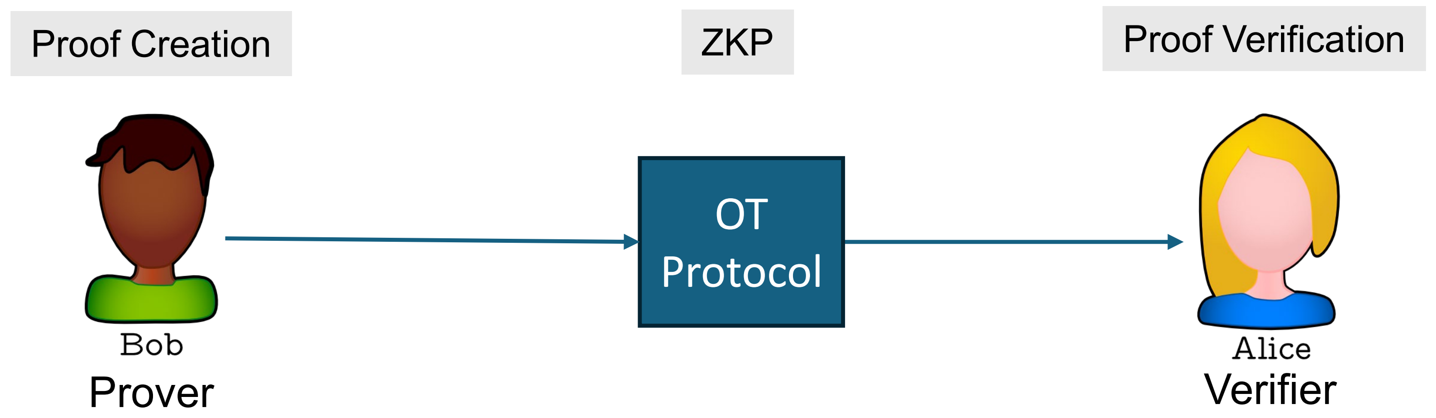


Garbled Table



Agreed Upon Circuit

Zero-Knowledge Proofs (ZKP)



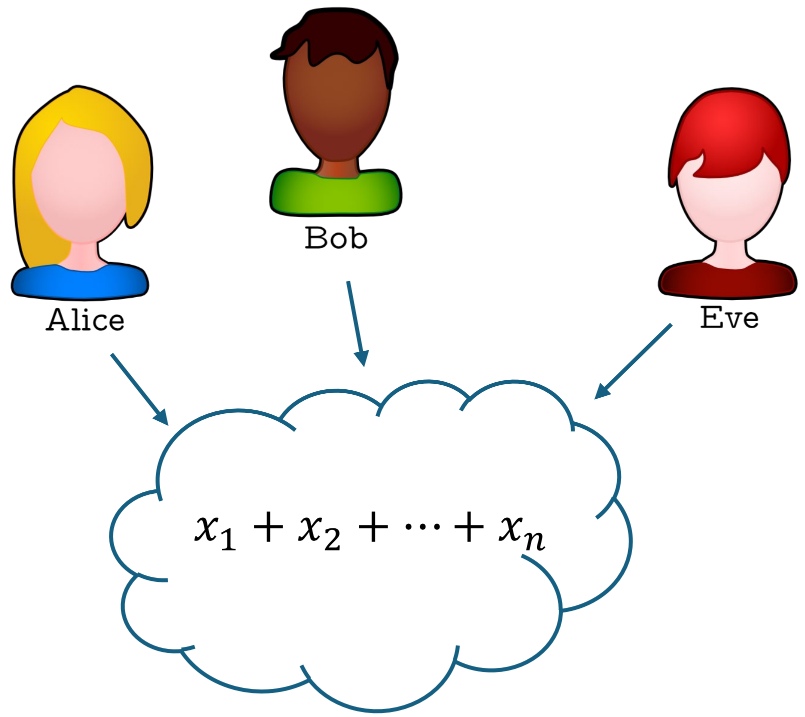
- Zero-knowledge proofs are cryptographic protocols that allow one party (the prover) to prove the validity of a statement to another party (the verifier) without revealing any information beyond the statement's validity.
- In SMPC, zero-knowledge proofs can be used to prove the correctness of computations or to demonstrate that certain conditions are met without revealing the underlying data.

Secure Multi-Party Operations

- Secure Multiparty Summation, Secure Multiparty Comparison, and Secure Function Evaluation – are “secure multiparty operations”.
  - These protocols are fundamental building blocks in the field of SMPC, enabling multiple parties to collaborate on computations while keeping their individual inputs private
  - By employing cryptographic techniques and protocols, such as secret sharing, homomorphic encryption, garbled circuits, and zero-knowledge proofs, parties can perform various operations securely without revealing sensitive information to each other.
  - Secure multiparty operations are essential for implementing privacy-preserving solutions in scenarios where parties wish to jointly analyze or process data while maintaining confidentiality and integrity.

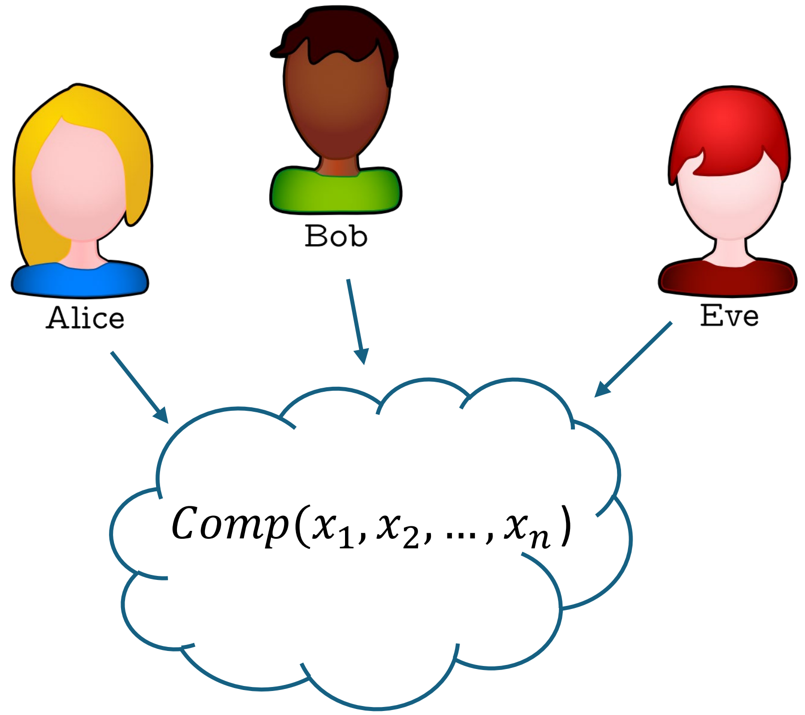
Secure Multiparty Summation

- Secure Multiparty Summation (SMS) is a cryptographic **protocol** that allows multiple parties to jointly compute the **sum** of their inputs while keeping those inputs private.
    - **Input Preparation**: may involve encoding their inputs using cryptographic techniques like secret sharing or HE.
    - **Addition Operations**: Parties perform **addition** operations on their encoded inputs using the agreed-upon cryptographic techniques.
    - **Combining Results**: After performing the addition operations, parties combine results of their computations.



Secure Multiparty Comparison

- Secure Multiparty Comparison (SMC) is a cryptographic **protocol** that allows multiple parties to **compare** their inputs while keeping those inputs private.
  - **Input Preparation**: may involve encoding their inputs using cryptographic techniques like secret sharing or HE.
  - **Addition Operations**: Parties perform **comparison** operations on their encoded inputs using cryptographic protocols such as oblivious transfer or secure garbled circuits.
  - **Combining Results**: The comparison results indicate whether the inputs are equal or if one input is greater than, less than another.



SMPC Implementation Tools

- MPyC (Multiparty Computation in Python)
- Sharemind
- SCALE-MAMBA
- SEAL
- ABY

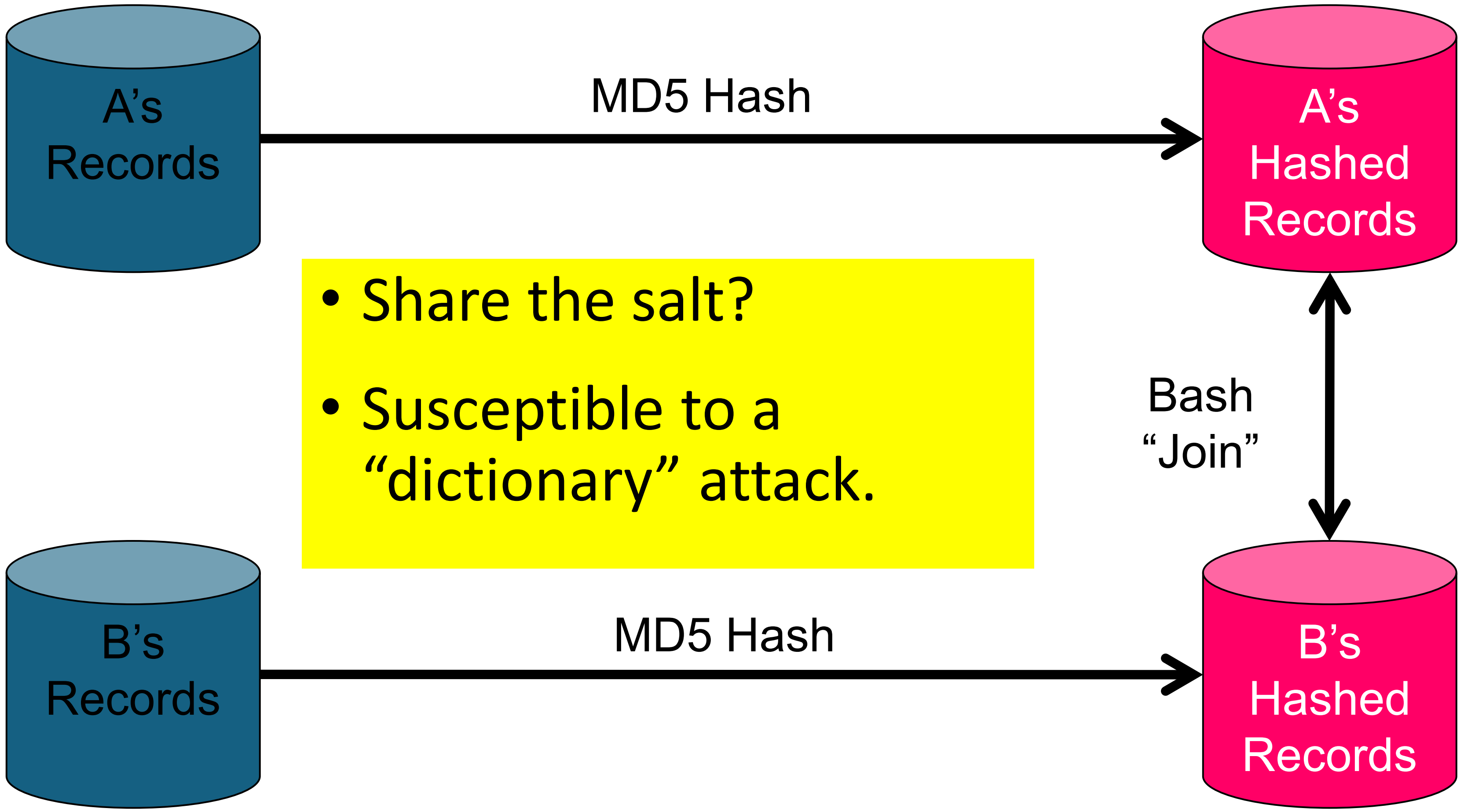
Hashed IDs

- ShanghaiTech could use MD5 for hashing medical record numbers

$$\mathrm{MD5}(\mathrm{mrn}, \mathrm{salt}) = \mathrm{pseudonym}$$

- Now, let’s say ShanghaiTech wants to compare its patient population with Shuguang Hospital down the street

Multiple Locations (Hash & Bash)

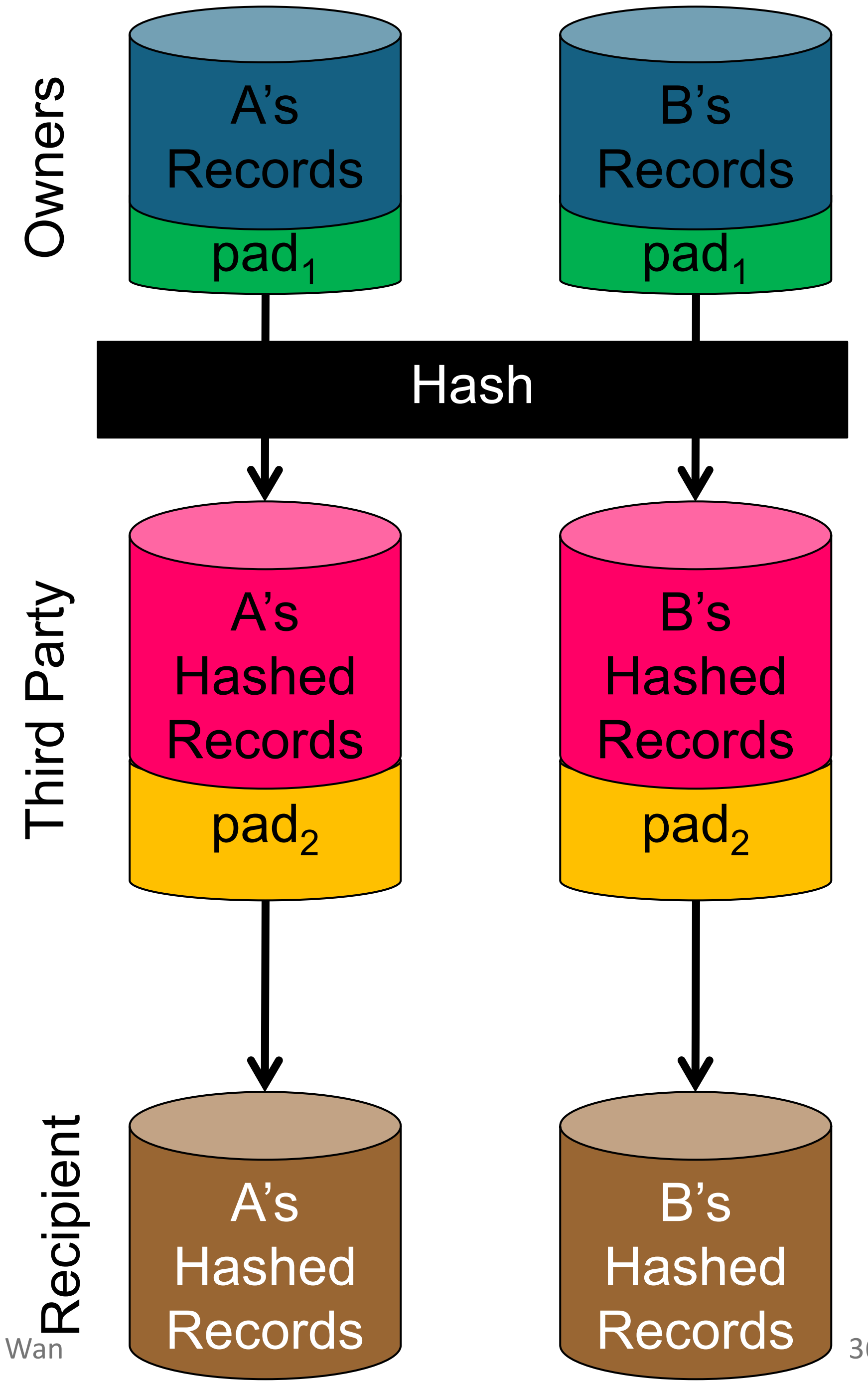


- Share the salt?
- Susceptible to a "dictionary" attack.

Adapted from Dr Malin's slides.

Double Hashing

- Propose concept akin to "re-encryption"
- Use 2 pads (*introduced before hashing*)
  - First pad used by all data senders
  - Second pad applied by the recipient
- Recommendation: Entity in charge of pad management should not
  - be a data sender or user
  - be given the hashed records



C. Quantin et al. Anonymous statistical methods versus epidemiology. International Journal of Medical Informatics. 2000; 60: 177-183.

Claim: Hash Codes Fail HIPAA

- "Comment: Several commenters who supported the creation of de-identified data for research based on removal of facial identifiers asked if a keyed-hash message authentication code (HMAC) can be used as a re-identification code even though it is derived from patient information, because it is not intended to re-identify the patient and it is not possible to identify the patient from the code. The commenters stated that use of the keyed-hash message authentication code would be valuable for research, public health and bio-terrorism detection purposes where there is a need to link clinical events on the same person occurring in different health care settings (e.g., to avoid double counting of cases or to observe long-term outcomes)….

  Response: The HMAC does not meet the conditions for use as a re-identification code for de-identified information. It is derived from individually identified information and it appears the key is shared with or provided by the recipient of the data in order for that recipient to be able to link information about the individual from multiple entities or over time. Since the HMAC allows identification of individuals by the recipient, disclosure of the HMAC violates the Rule….

  …The HMAC methodology, however, may be used in the context of the limited data set, discussed below. The limited data set contains individually identifiable health information and is not a de-identified data set."

Zero Knowledge?

- Zero knowledge proof requires interaction between
  - Prover *P*
  - Verifier *V*
- *P* convinces *V* about a statement without revealing information about how to prove the statement

Zero Knowledge: Example

- Remember the basis of RSA?
- Assumption: arithmetic modulo *n*

  n = *pq*, where *p* and *q* are primes

  Factoring *n* is intractable

- Finding the *square root mod n* is equivalent to factoring *n*
  - In other words, if you have an algorithm *a* that can find a *square root of the number mod n*, then you can use *a* to factor *n*

- Proof uses multiple rounds of interaction that show
  - *P* knows a square root of a published number
  - without revealing new information about the square root

Zero Knowledge: Example

- It is known there is a square root of the number; i.e., the “quadratic residue” (QR)
- When there is an integer $0 < s < n$, such that

$$s^2 \equiv d \bmod n$$

  d is the quadratic residue
  - Note: $d = 0$ is usually excluded
- Alternatively: $d \equiv s^2 \bmod n$
- $6^2 \equiv 6 \bmod 10$
- $7^2 \equiv 9 \bmod 10$
- Hard to tell if a random number x is a quadratic residue, without knowing a factorization of n

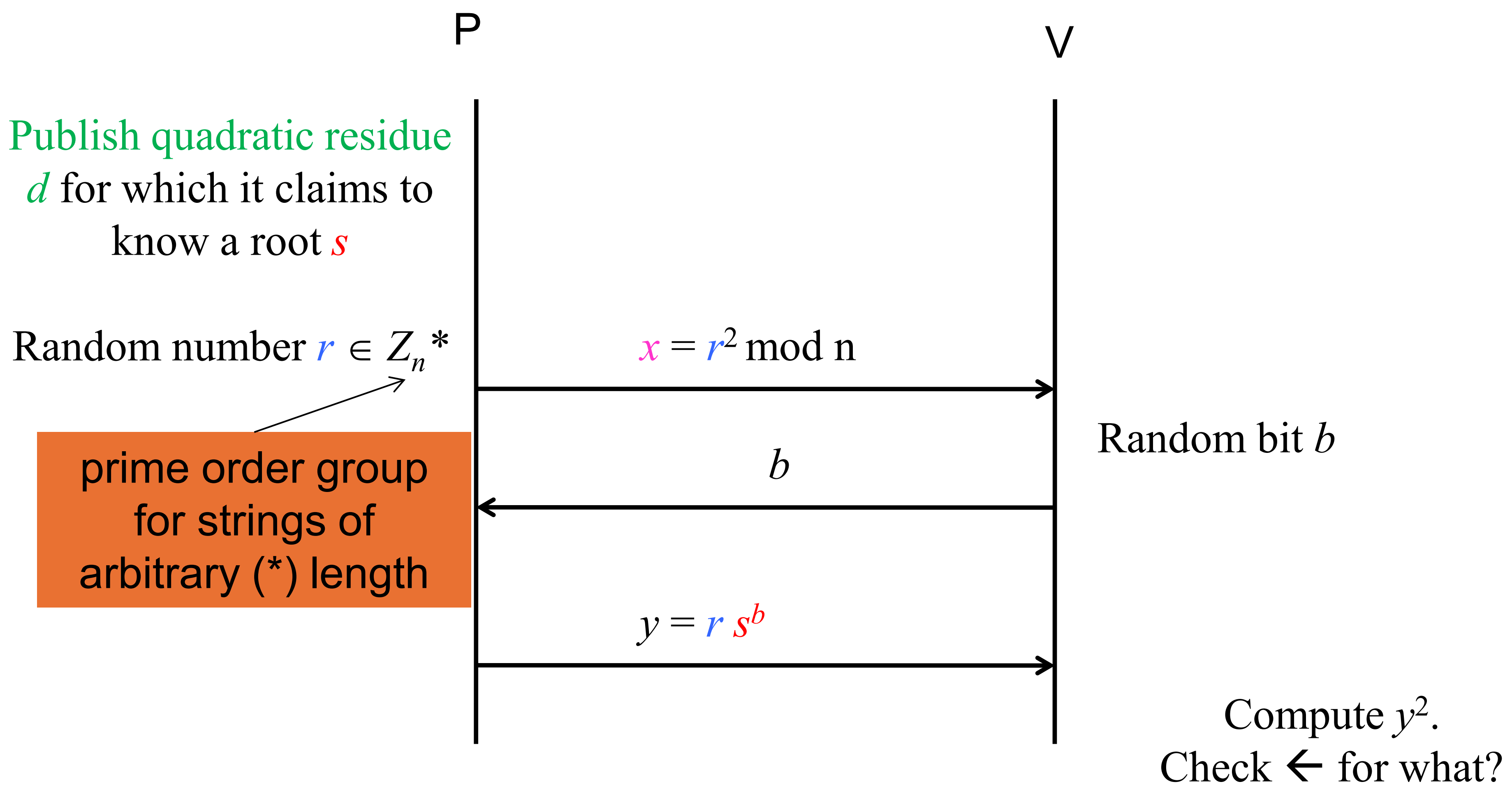
Zero Knowledge: Example

- It is known there is a square root of the number; i.e., the “quadratic residue” (QR)
- The factors of the *mod n* may be secret

- *P* publishes the quadratic residue *d* for which he claims to know root *s*

- When *P* needs to prove its knowledge of *s* to *V, P* runs several rounds of interaction
- In each round, *P* chooses a new random number *r*
  - Sends $x = r^2$ mod *n* to *V*
- *V* chooses a random bit *b*
  - Sends *b* to *P*

Actions


P
V
Publish quadratic residue d for which it claims to know a root s
Random number $r \in Z_n{}^*$
prime order group for strings of arbitrary (*) length
$x = r^2 \bmod n$
Random bit b
b
$y = r\, s^b$
Compute y^2.
Check ← for what?

Proofs

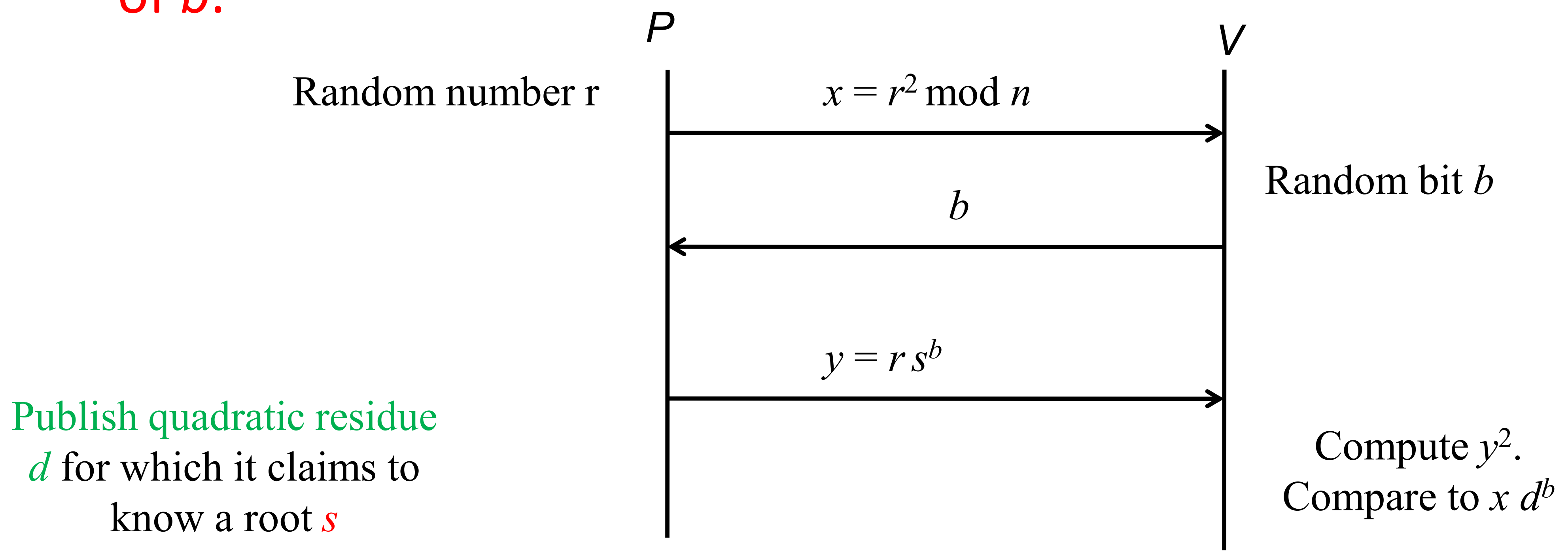
- Completeness
- Soundness
- Zero-Knowledge

Proofs

- Completeness: Every time
  - *d* is a *QR*
  - *V* is given $d \leftarrow s^2 \bmod n$
  - *P* and *V* follow the protocol
  - ***V* accepts with probability 1**

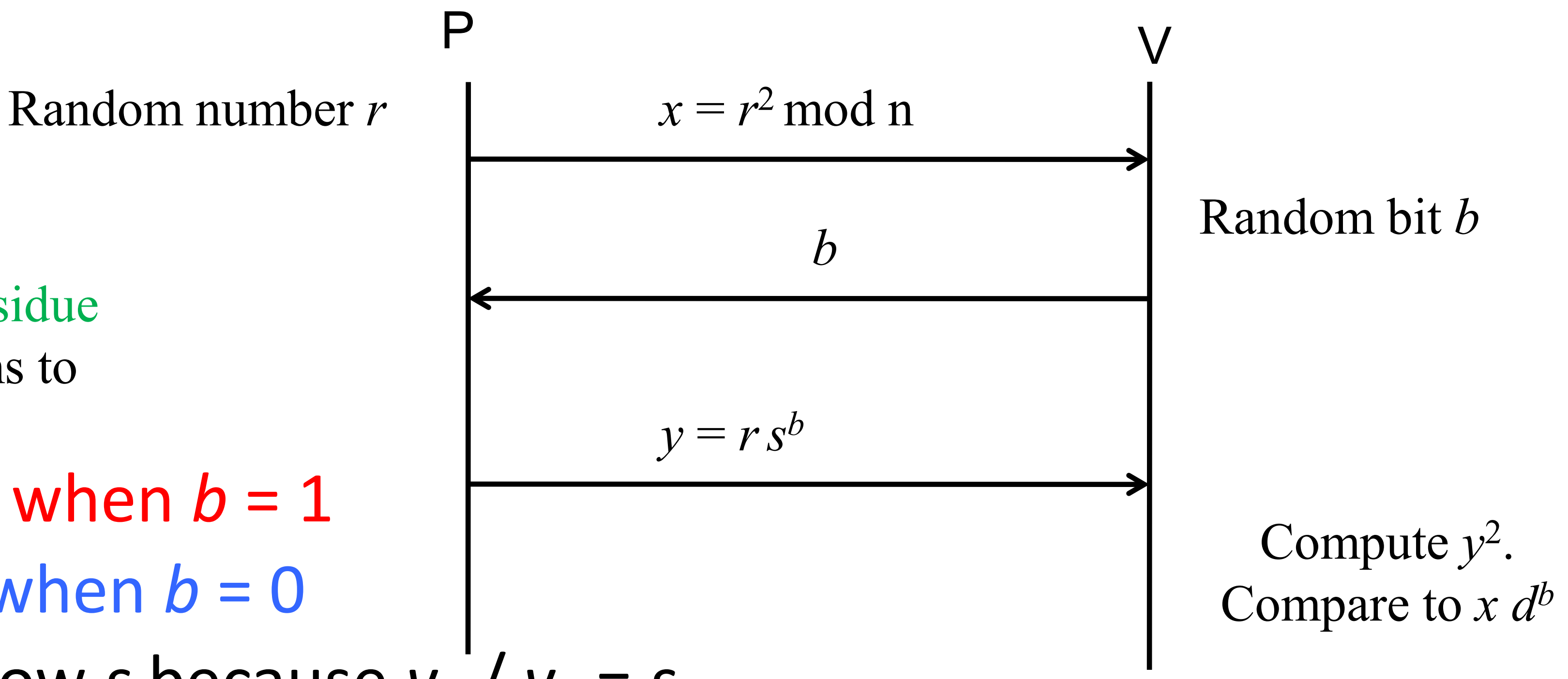
Claim 1: Completeness

- Only *P* can successfully complete the protocol for both possible values of *b*.



Claim 1: Completeness

- Only *P* can successfully complete the protocol for both possible values of *b*.



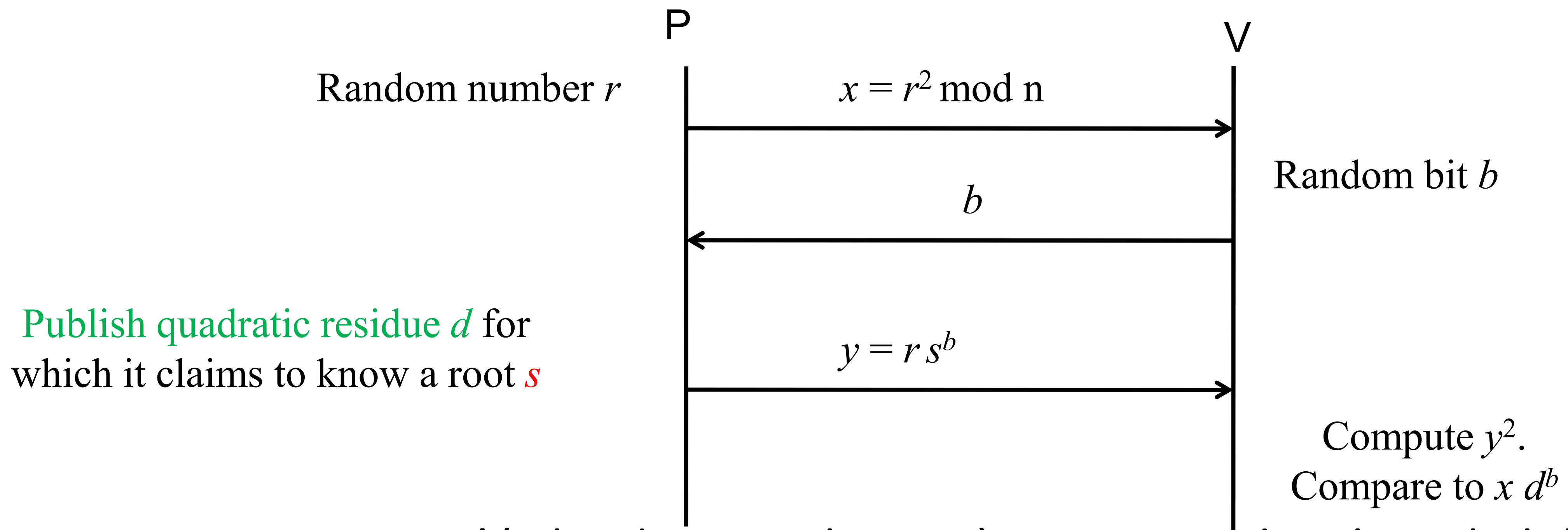
Publish quadratic residue d for which it claims to know a root s

- Know $y_1 = rs$ when $b = 1$
- Know $y_2 = r$ when $b = 0$
- Follows *P* know *s* because $y_1 / y_2 = s$

Proofs

- Soundness:
    - If d is not a quadratic residue,
    - then V will reject with probability ≥ 0.5

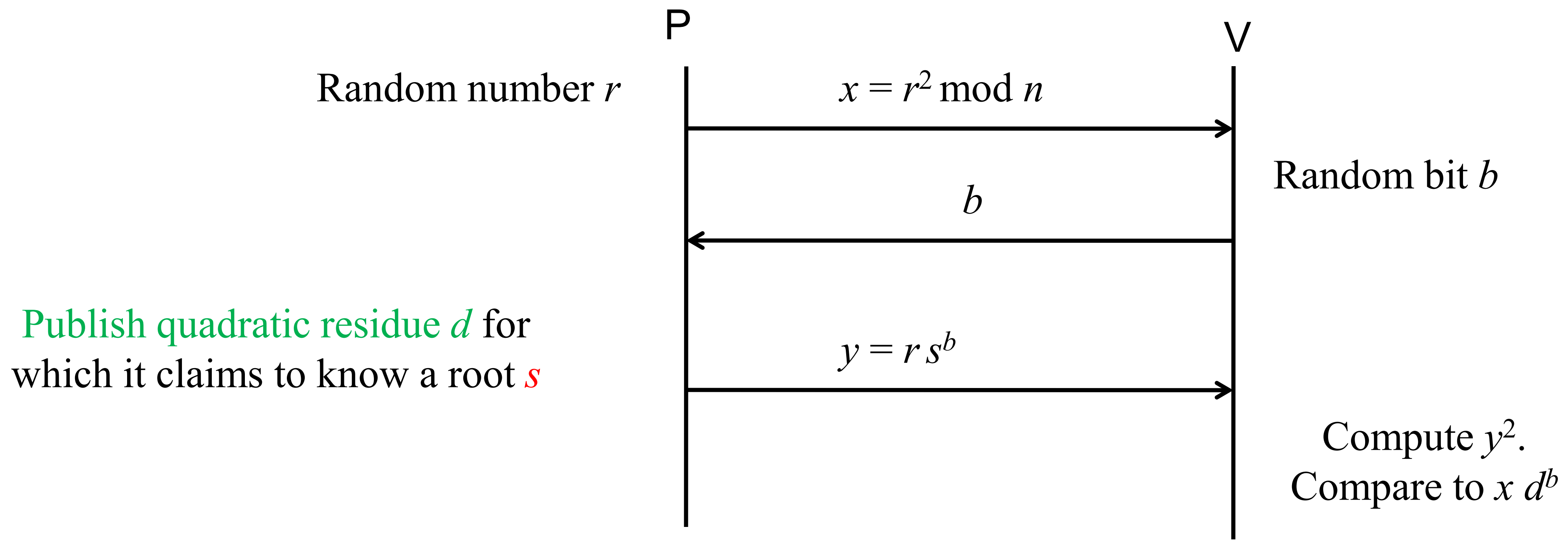
Claim 2


P
V
Random number r
$x = r^2 \bmod n$
Random bit b
b
Publish quadratic residue d for which it claims to know a root s
$y = r\,s^b$
Compute y^2. Compare to $x\,d^b$

- An imposter *P'* (who does not know *s*) can succeed with probability 0.5 each round.
  - If *P'* correctly guesses $b = 0$, it can follow the protocol and succeed
  - If *P'* incorrectly guesses $b = 1$, *P'* can generate x by choosing a random number t, such that $x = t^2 / d$

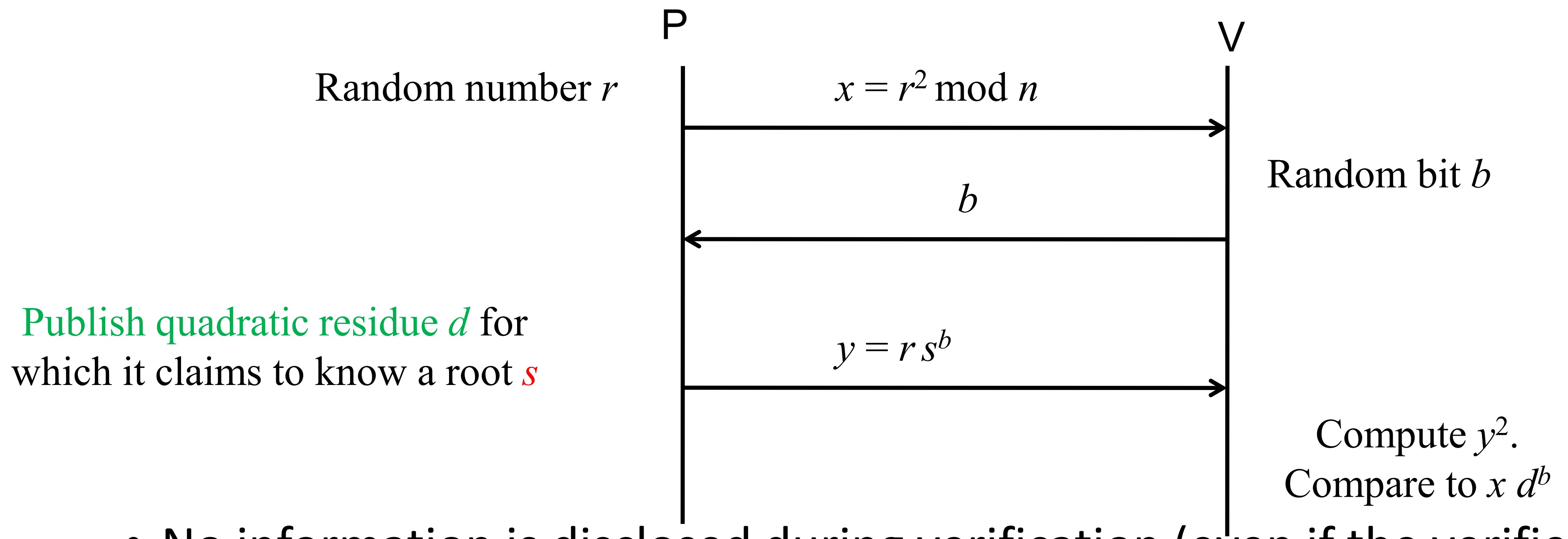
    … and the response is $y = t$.

Claim 3



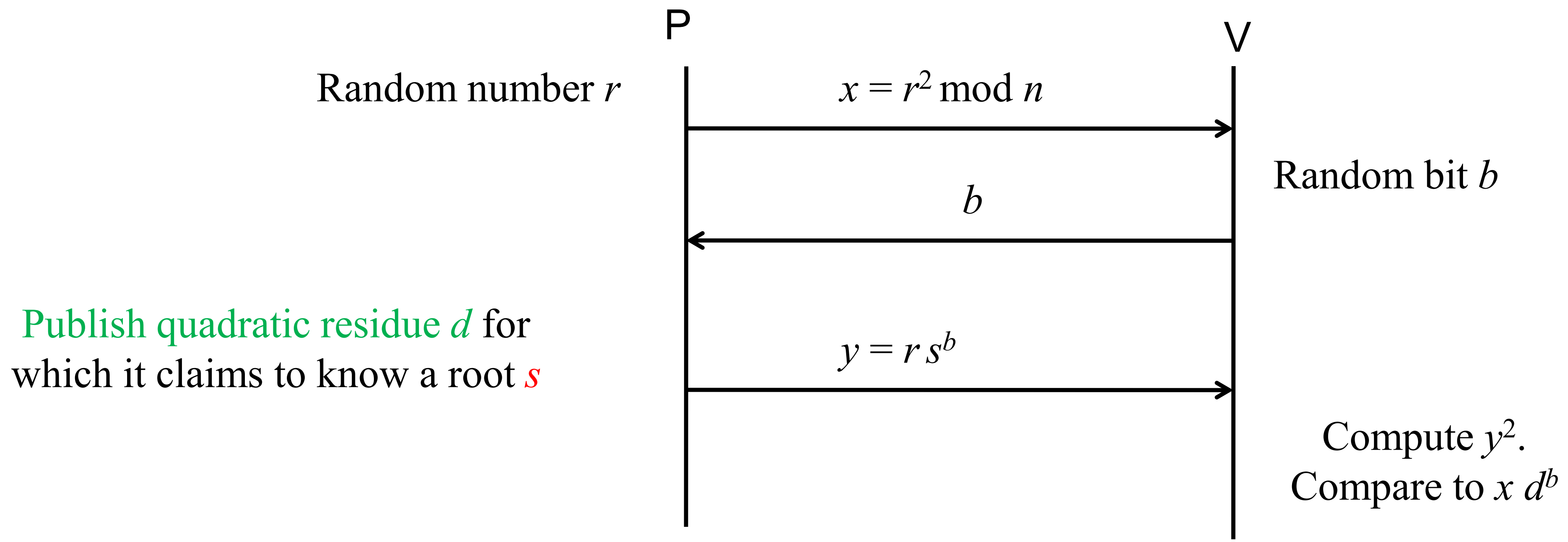
- No information is disclosed during verification (even if the verifier is cheating)

Claim 3


P
V
Random number r
$x = r^2 \bmod n$
Random bit b
b
Publish quadratic residue d for which it claims to know a root s
$y = r\,s^b$
Compute y^2. Compare to $x\,d^b$

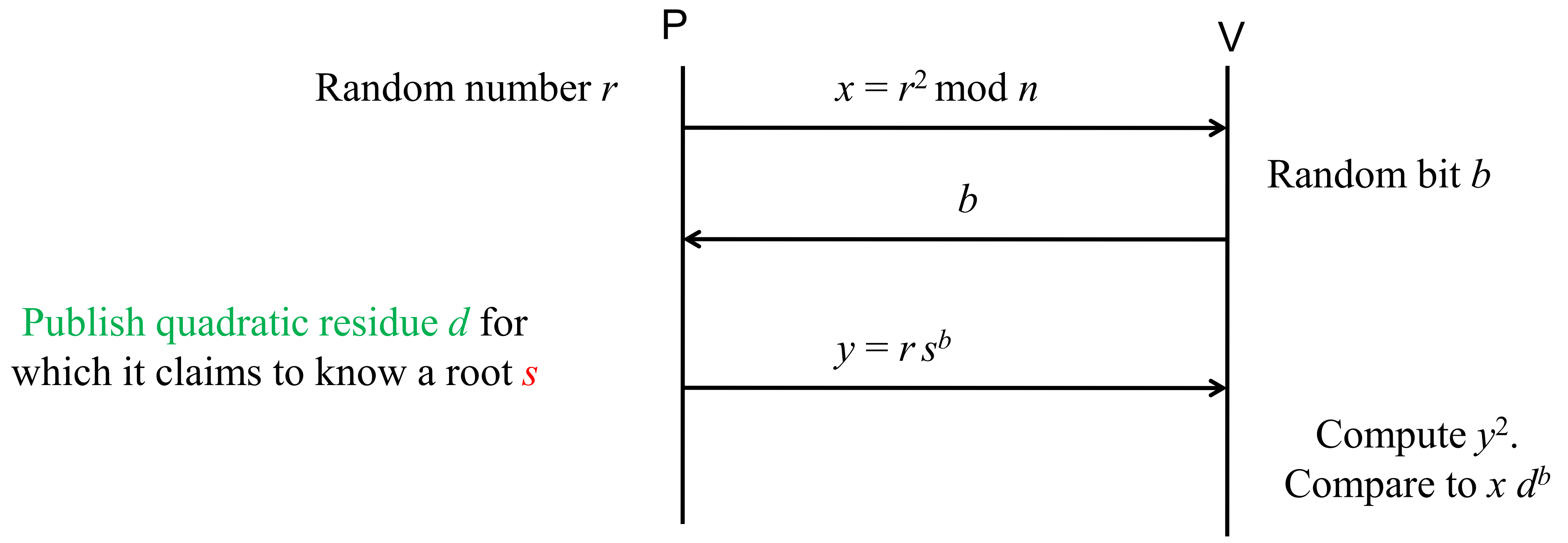
- No information is disclosed during verification (even if the verifier is cheating)
  - Consider an eavesdropper *Eve* on the line
  - When $b = 0$, *Eve* sees a random number r and its square x
  - When $b = 1$, *Eve* sees the numbers rs and $x = (rs)^2 / d$
  - But these do not imply s

Claim 3


P
V
Random number r
$x = r^2 \bmod n$
Random bit b
b
Publish quadratic residue d for which it claims to know a root s
$y = r\,s^b$
Compute y^2. Compare to $x\,d^b$

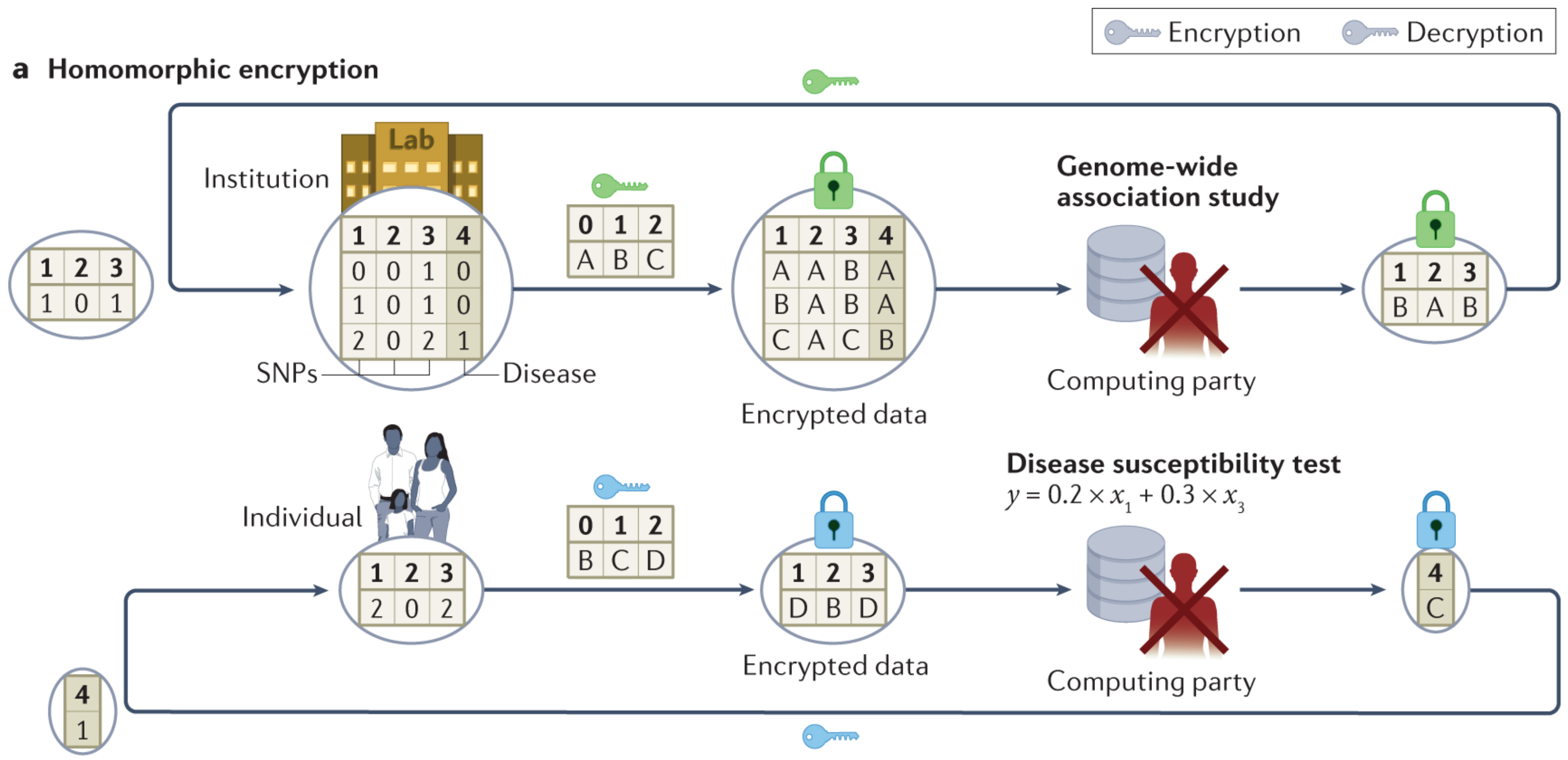
- No information is disclosed during verification (even if the verifier is cheating)
  - More formally, we would use a “simulator” to run both sides of the protocol
  - With advanced information - the value of the random bit – can simulate the protocol without knowledge of *s*

Probabilistic


P
V
Random number r
$x = r^2 \bmod n$
Random bit b
b
Publish quadratic residue d for which it claims to know a root s
$y = r\,s^b$
Compute y^2. Compare to $x\,d^b$

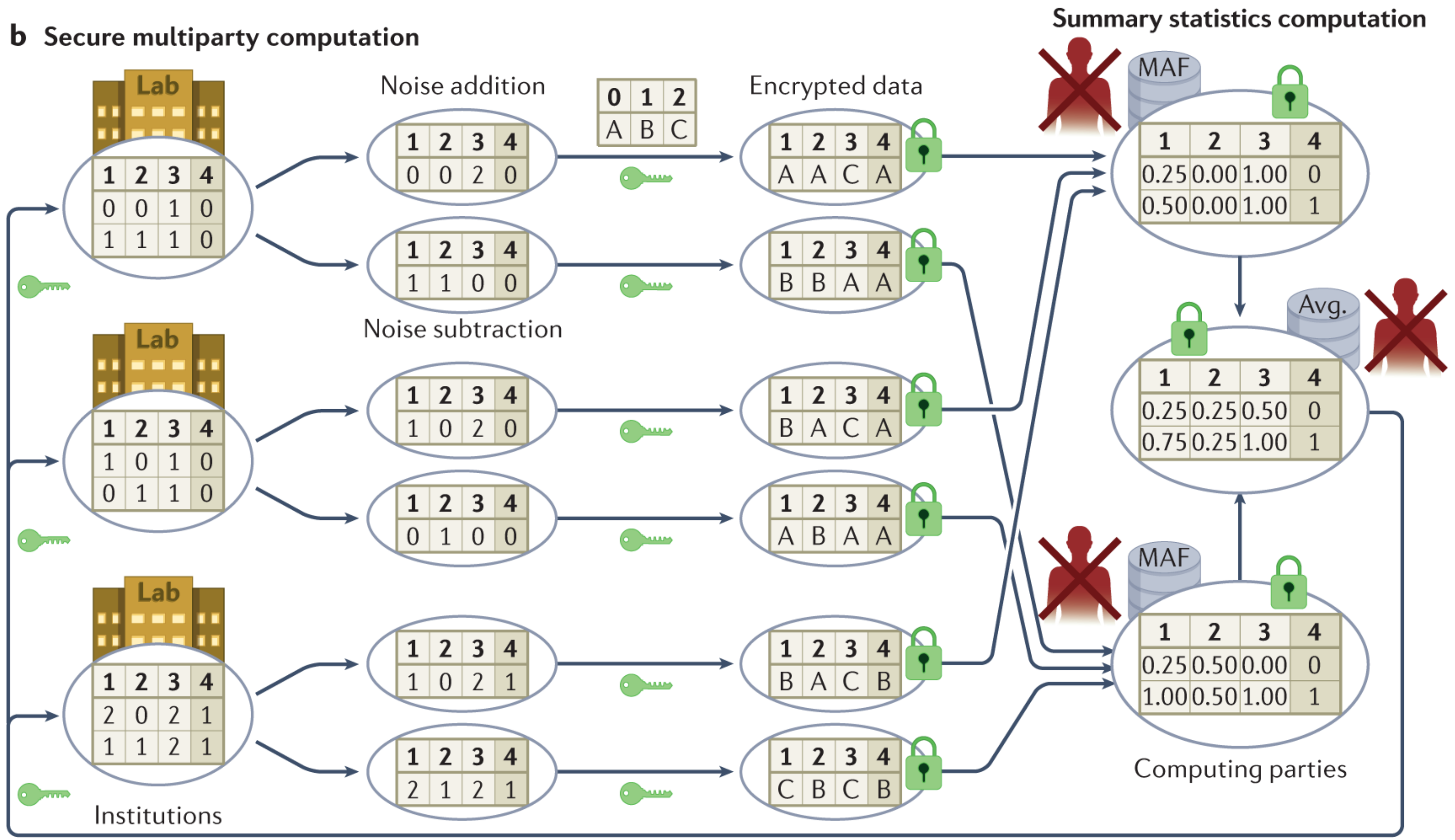
- *Conceptually, similar to the partial key switch detection algorithm*
- Each round, there is a 0.5 probability *P* does not know *s*
- After 10 rounds, there is ~0.00098 probability *P* does not know *s*

Applications in Genomic Privacy



Wan Z, Hazel JW, Clayton EW, Vorobeychik Y, Kantarcioglu M, Malin BA. Sociotechnical safeguards for genomic data privacy. Nature Reviews Genetics. 2022 Jul;23(7):429-45.

Applications in Genomic Privacy



Wan Z, Hazel JW, Clayton EW, Vorobeychik Y, Kantarcioglu M, Malin BA. Sociotechnical safeguards for genomic data privacy. Nature Reviews Genetics. 2022 Jul;23(7):429-45.

Readings for the Next Week

- 1. N/A

- Optional
  - ❑ 2. Li T, Sahu AK, Talwalkar A, Smith V. **Federated learning: Challenges, methods, and future directions**. *IEEE signal processing magazine*. 2020 May 1;37(3):50-60. https://doi.org/10.1109/MSP.2020.2975749
  - ❑ 3. Yan C, Yan Y, Wan Z, Zhang Z, Omberg L, Guinney J, Mooney SD, Malin BA. **A multifaceted benchmarking of synthetic electronic health record generation models**. *Nature communications*. 2022 Dec 9;13(1):7609. https://doi.org/10.1038/s41467-022-35295-1

Feedback Survey

- One thing you learned or felt was valuable from today's class & reading
- Muddiest point: what, if anything, feels unclear, confusing or "muddy"
- https://www.wjx.cn/vm/hX0mIro.aspx