CRYPTOGRAPHIC FILEMATCHER

Ngoc N. Tran (ngoc@underlandian.com)

ABSTRACT

- The idea is to compare the hashes securely.
- Comparison is essentially secure 2-party computation.

A LITTLE LESS ABSTRACT

- Secure channel is realized with encryption and verification

  (latter is on the users' own)

- Secure 2-party computation (set intersection) is done with a partially homomorphic encryption called

  The Paillier Cryptosystem.

FILE REPRESENTATION

- Hashes are done with SHA3-512.
- Stored as arbitrary length integers.
- Sent as a polynomials of said roots.

$$f(x) = (x - h_1)(x - h_2)\ldots(x - h_n)$$

PROTOCOL (1)

- Sender gets said polynomial
- Sender generates public/secret keypair
- Sender encrypts polynomial's coefficients with public key and send f'(x).

PROTOCOL (2)

- Receiver computes own's hashes
- Evaluates the received f'(x) at said hashes
- Randomize each result and send back

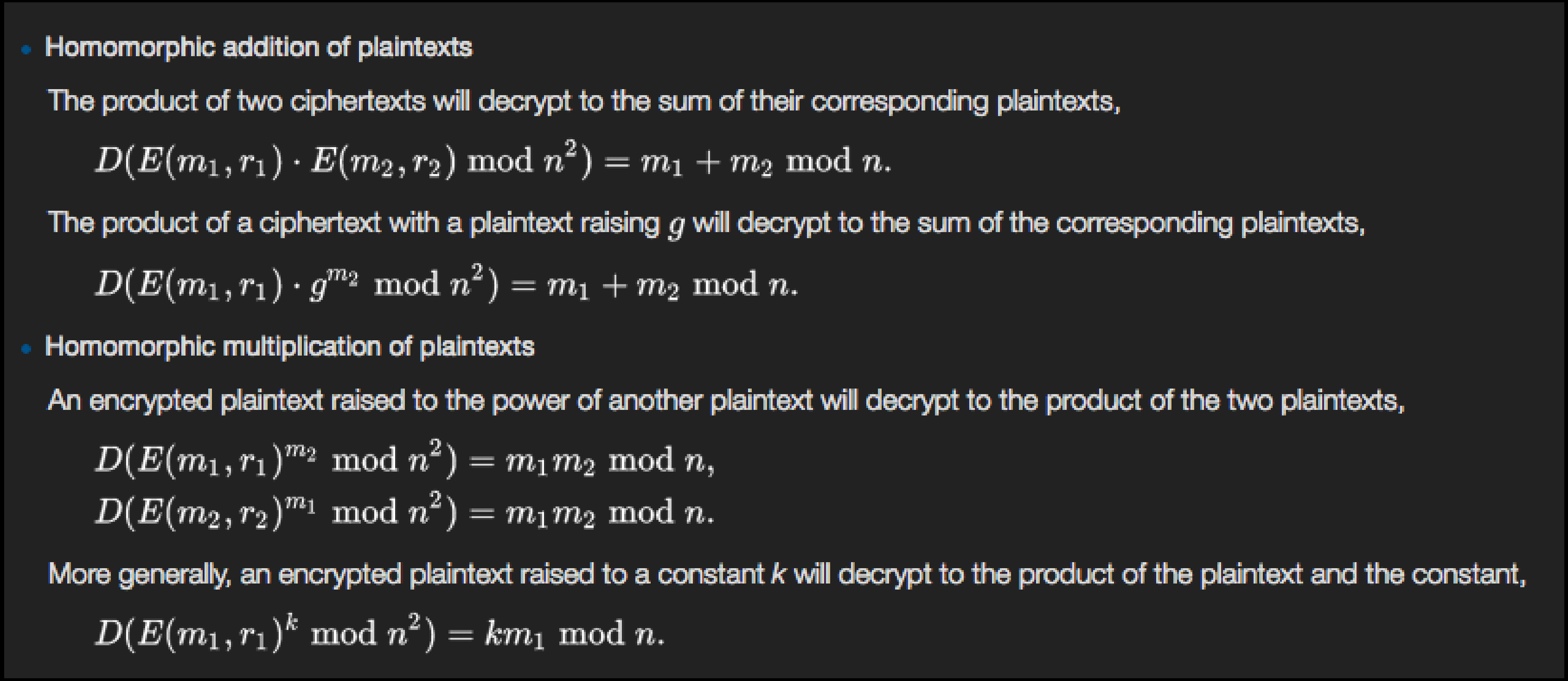
$$r_i f'(h_i') + h_i'$$

PROTOCOL (3)

- Sender decrypts the received numbers with his own private key
- The decrypted numbers matching his own hashes are the common files.
- Swap roles, rinse and repeat.

PAILLIER CRYPTOSYSTEM

- The technical details are omitted.
- Property: homomorphic with linear operations.



- Homomorphic addition of plaintexts

  The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts,

  $$D(E(m_1, r_1) \cdot E(m_2, r_2) \bmod n^2) = m_1 + m_2 \bmod n.$$

  The product of a ciphertext with a plaintext raising g will decrypt to the sum of the corresponding plaintexts,

  $$D(E(m_1, r_1) \cdot g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n.$$

- Homomorphic multiplication of plaintexts

  An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts,

  $$D(E(m_1, r_1)^{m_2} \bmod n^2) = m_1 m_2 \bmod n,$$
  $$D(E(m_2, r_2)^{m_1} \bmod n^2) = m_1 m_2 \bmod n.$$

  More generally, an encrypted plaintext raised to a constant *k* will decrypt to the product of the plaintext and the constant,

  $$D(E(m_1, r_1)^k \bmod n^2) = k m_1 \bmod n.$$

WHY IT WORKS

- Homomorphic: f(x) = 0 <-> f'(x) = 0 (proof omitted)
- Post-decrypt hash comparison:

$$r_i f'(h'_i) + h'_i = \begin{cases} h'_i & \text{if } f'(h'_i) = 0 \\ r'_i & \text{otherwise.} \end{cases}$$

- Encrypted coefficients: no information about sender's
- Randomized results: no information about receiver's

SECURITY

- Zero knowledge (last slide)
- Secure channel (Encryption and Signature)

CONCERNS

- Possible symmetric protocol replacement
- Possible Paillier implementation weakpoint
- Possible getting out of range of n^2 (not our case)
- SHA3-512 is slow.

THAT'S ALL FOLKS!

Any questions?