Supporting Less-Than Queries on Encrypted Data using Multi-Server Secret Sharing and Practical Order-Revealing Encryption

Nate Chenette

ICERM conference on Encrypted Search

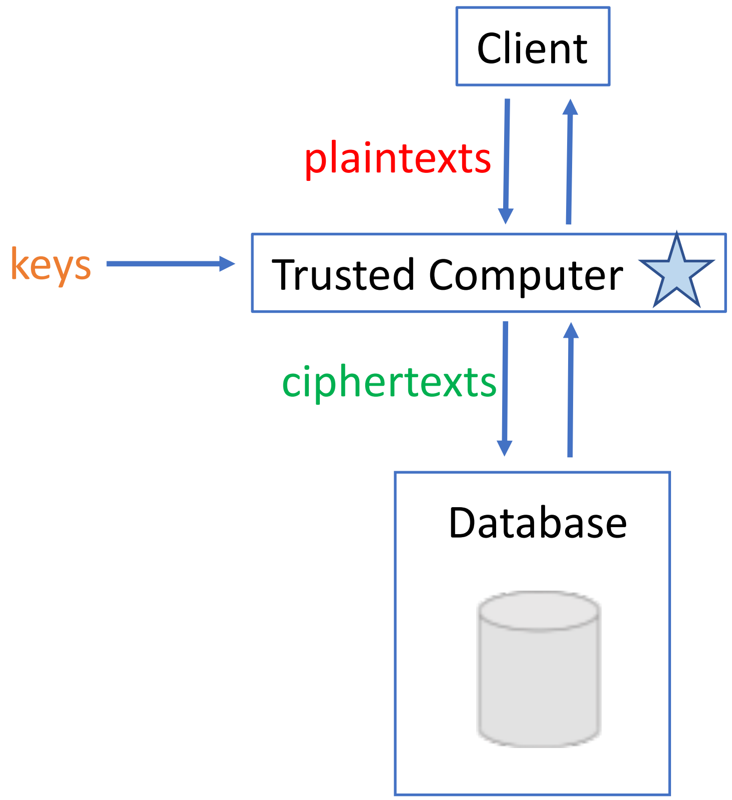
June 12, 2019

Project Background

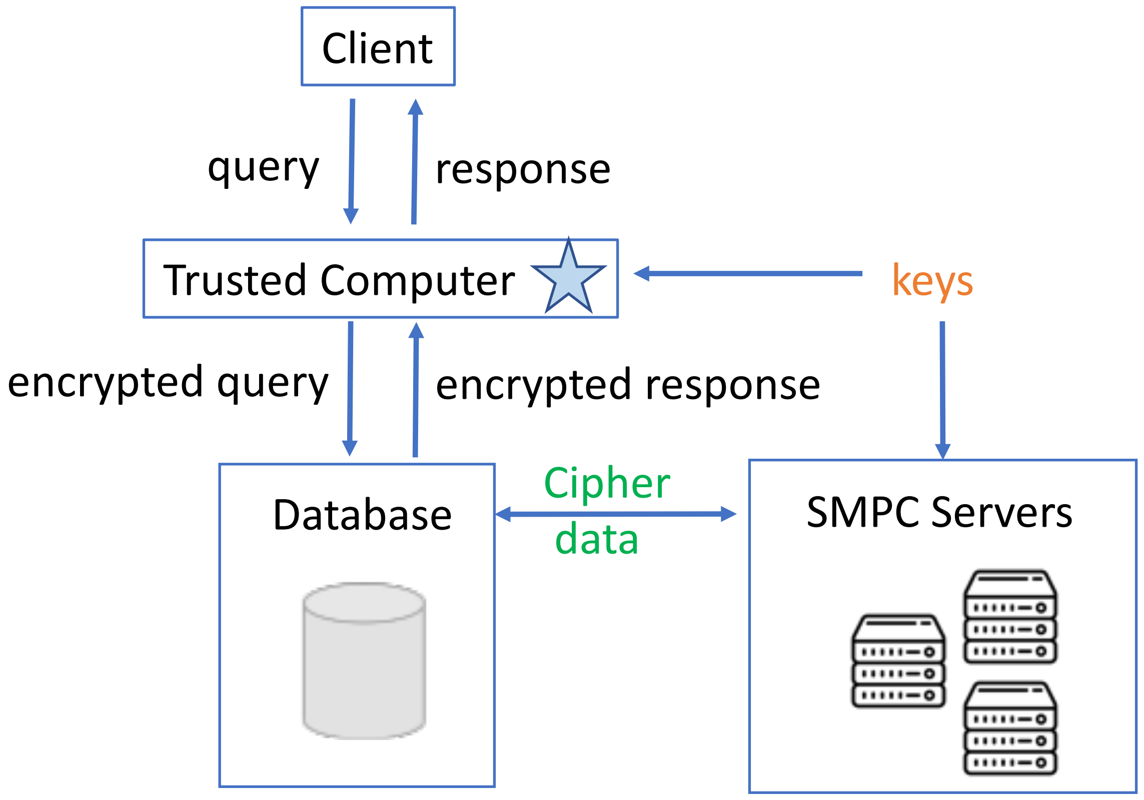
- Baffle Inc. https://baffle.io/
  - Goal: implement fully-fledged database server that provides a strong level of security
    - "Baffle provides an advanced data protection solution that protects data in memory, in process and at-rest to reduce insider threat and data theft risk."
  - Many of their schemes implement searchable encryption!
  - Security model: multiple servers, assume **only one** is compromised by an active adversary
  - Protect as much information as possible, while supporting various query types (addition, equality, comparison)

- My role as a consultant: evaluate schemes for comparison operations on encrypted data, specifically involving order-revealing encryption

Baffle System Architecture



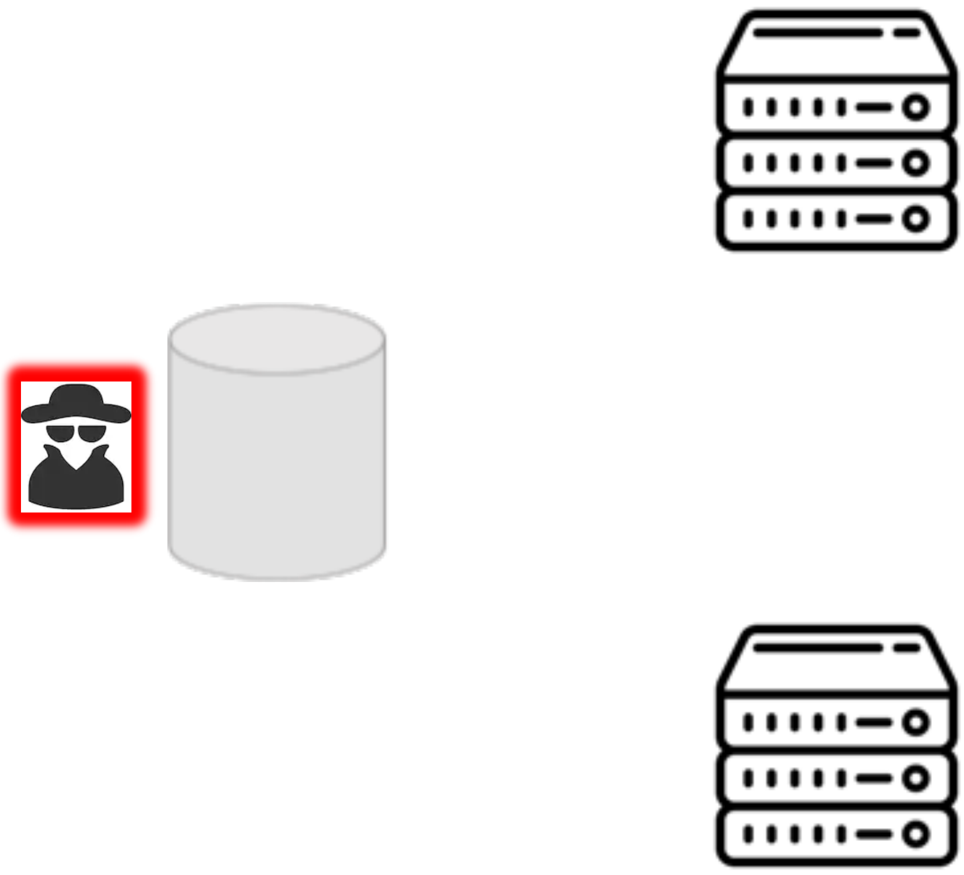
Encryption/Decryption



Query Support

Functionality and Security Model

- Multiple servers
- Respond to queries via an efficient collaborative protocol
  - Addition
  - Equality
  - Less-than
- Assume an intruder can only compromise **one server at a time**
- Characterize (and minimize) the leakage



Baffle Encryption (& Authentication)

- Uses a pseudorandom function *F* and a MAC *M*
- Secret sharing, Encrypt-then-MAC

Plaintext d

Encryption key k
Authentication key k_a

Trusted Computer

- Choose random nonce n
- Compute pad by running PRF on nonce with enc. key:

$$p = F(k, n)$$

- Compute ciphertext c by subtracting* plaintext from tag:

$$c = p - d = F(k, n) - d$$

- Compute MAC:

$$m = M(k_a, n, c)$$

(n, c, m)

*All quantities and operations occur in some finite commutative ring, e.g., the integers mod 256

Baffle Authenticated Decryption

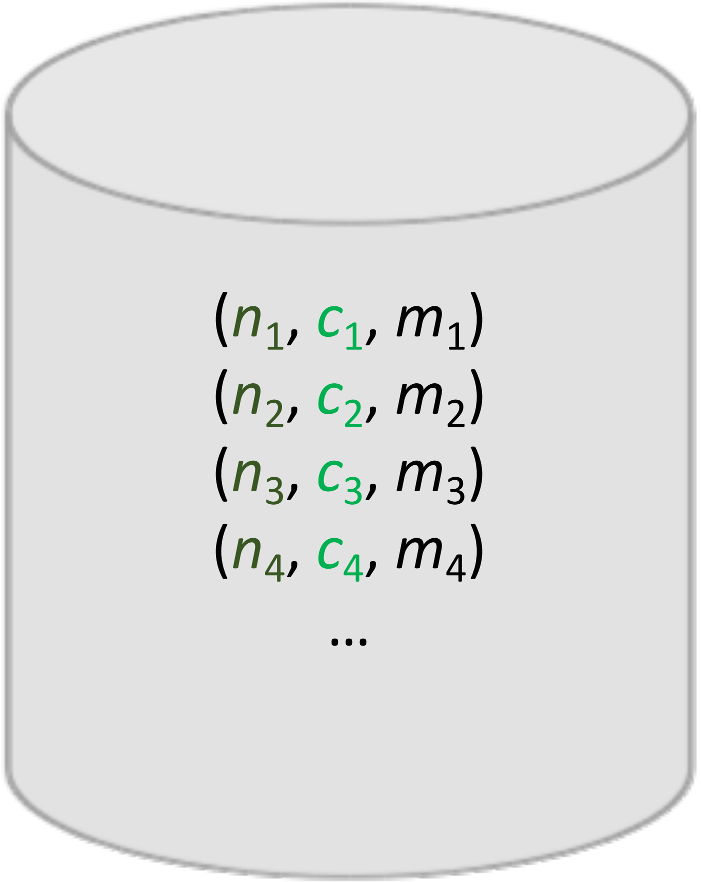
- Recall $c = F(k, n) - d$
- So, $d = F(k, n) - c$

- To decrypt (n, c, m):
  - Check the MAC: Verify $m == M(k_a, n, c)$
  - If so, re-compute the pad $F(k, n)$ from the nonce and subtract.

$$d = F(k, n) - c$$

Trusted Computer

Database View



Baffle Encrypted Addition

$c_1 = F(k, n_1) - d_1$
$c_2 = F(k, n_2) - d_2$

Correctness:

$$c_3 = S + c_1 + c_2$$
$$= F(k, n_3) - F(k, n_1) - F(k, n_2) + c_1 + c_2$$
$$= F(k, n_3) - (d_1 + d_2)$$

Query from client via trusted computer: ADD (n_1, c_1, m_1) to (n_2, c_2, m_2)

Encryption key k

Database

- Compute ciphertext:
  $c_3 = S + c_1 + c_2$
- Cipher tuple of sum is
  $(n_3, c_3, 0^*)$

n_1, n_2 →

← n_3, S

Server 1

- Choose random nonce n_3
- Compute pad by running PRF on nonce:
  $p = F(k, n_3)$
- Compute pad difference
  $S = p - F(k, n_1) - F(k, n_2)$

*DB can't compute the MAC, but the trusted computer could before returning the tuple

Security notes:

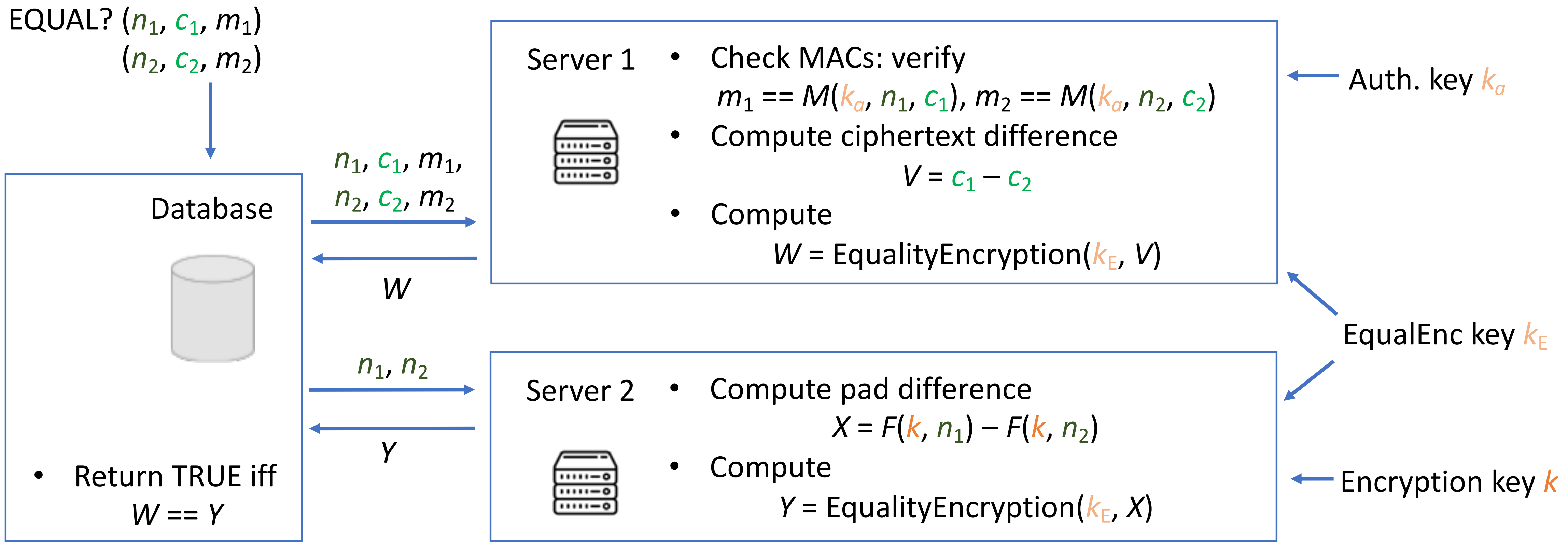
- All of Server 1's information is independent of plaintext data!
- Database doesn't have k, so can't uncover pads from (independent) nonces.

Baffle Encrypted Equality

$c_1 = F(k, n_1) - d_1$
$c_2 = F(k, n_2) - d_2$

X V

Correctness: $W == Y$ iff $V == X$ iff $d_1 == d_2$



EqualityEncryption preserves equality; details explained on next page

Security notes:

- Again, plaintext-dependent data (at Database & Server 1) has been separated from the ability to decrypt (Server 2).
- The ability of the Database to discover sensitive information is dependent on the security of EqualityEncryption.

EqualityEncryption

- In principle, could be any equality-revealing encryption such as deterministic encryption [Bellare, Boldyreva, O'Neill 2007]
- As the EqualEnc key is new for each Equality query, Baffle gets away with a simple affine encryption. Use the key k_E to generate invertible multiplier α and shift β, and compute

$$\text{EqualityEncryption}(k_E, V) = \alpha V + \beta$$

- Since α is invertible, EqualityEncryption(k_E, V) == $\alpha V + \beta$ == $\alpha X + \beta$ == EqualityEncryption(k_E, X) if and only if $V == X$.

Encrypted Comparison – First Try

$c_1 = F(k, n_1) - d_1$
$c_2 = F(k, n_2) - d_2$

LESSTHAN? (n_1, c_1, m_1)
(n_2, c_2, m_2)

Database

- Return result of ~~ORE-LessThan~~(W, Y)

$n_1, c_1, m_1,$
n_2, c_2, m_2

W

n_1, n_2

Y

Server 1

- Check MACs: verify $m_1 == M(k_a, n_1, c_1)$, $m_2 == M(k_a, n_2, c_2)$
- Compute ciphertext difference
  $V = c_1 - c_2$
- Compute
  $W = \cancel{\text{OrderRevealingEncryption}}(k_L, V)$

Server 2

- Compute pad difference
  $X = F(k, n_1) - F(k, n_2)$
- Compute
  $Y = \cancel{\text{OrderRevealingEncryption}}(k_L, X)$

Auth. key k_a

ORE key k_L

Encryption key k

Correctness (?):

$$d_1 - d_2 = (F(k, n_1) - c_1) - (F(k, n_2) - c_2)$$
$$= (F(k, n_1) - F(k, n_2)) - (c_1 - c_2)$$
$$= X - V$$

So $d_1 < d_2$ iff $d_1 - d_2 < 0$ iff $X - V < 0$ iff $X < V$, which matches the result of ORE-LessThan(W, Y)

Wrong!!

Dealing With Signs

- For simplicity, assume plaintexts are ASCII characters, i.e., in Z_{128}
- Clarification: arithmetic is performed in Z_{256}, represented using (two's complement) signed bytes, i.e. taking values in the range [–128,127].

Claim from previous page: $d_1 < d_2$ iff $d_1 - d_2 < 0$ iff $X - V < 0$ iff $X < V$

True, since d_1 and d_2 are assumed to be ASCII characters in the range [0,127] while $d_1 - d_2$ is in the range [–127,127].

True, since $d_1 - d_2 = X - V$

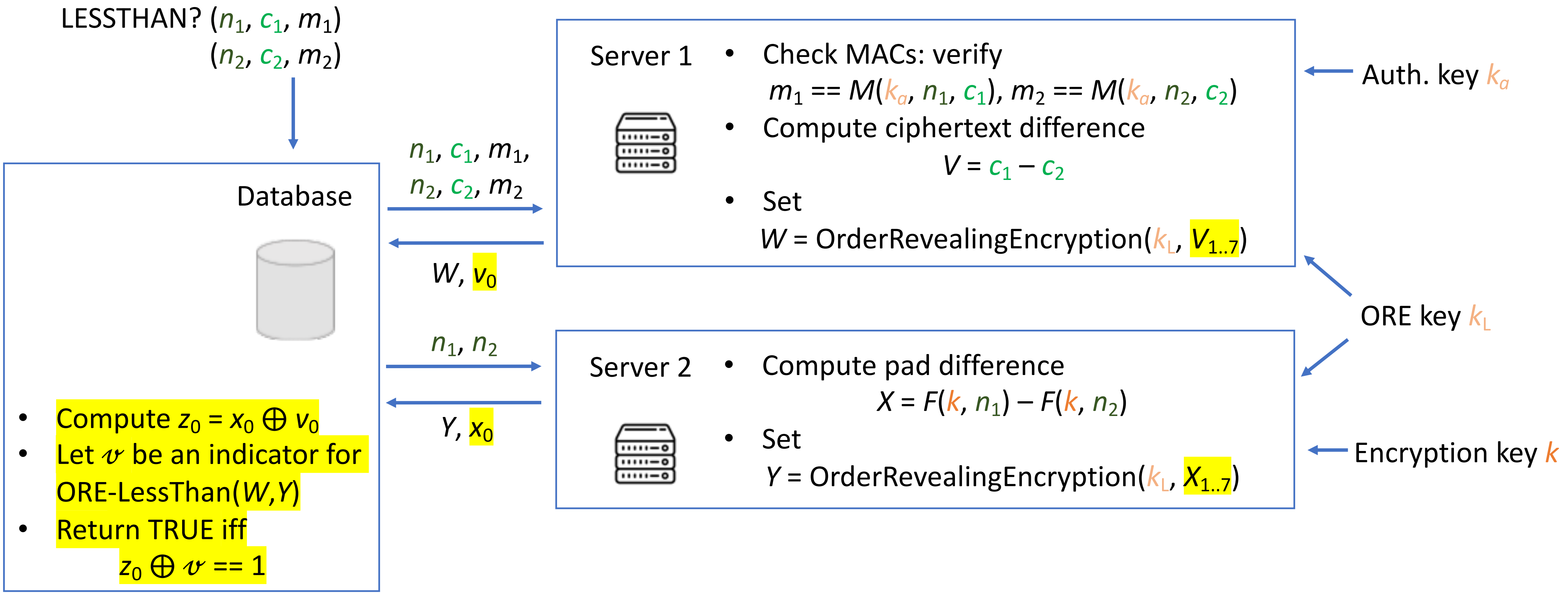
False, because of modularity of the subtraction. For example, $X = 100$, $V = -30$ gives
$X - V = -126 < 0$;
$X \geq V$.

Solution:

- Let $z_0 = x_0 \oplus v_0$ be an indicator for whether the sign bits of X and V differ.
- Let $\mathscr{v}$ be an indicator for whether $X_{1..7} < V_{1..7}$, where we are comparing the non-signed parts of X and V.
- Then $X - V < 0$ iff $z_0 \oplus \mathscr{v} == 1$.

Encrypted Comparison – Corrected

$c_1 = F(k, n_1) - d_1$
$c_2 = F(k, n_2) - d_2$



Baffle Implementation of Comparison

- For OrderRevealingEncryption(k_L,·), use a variant of the "Practical Order-Revealing Encryption" scheme [Chenette, Lewi, Weis, Wu 2016]
  - Leakage: order of $V_{1..7}$ and $W_{1..7}$, and the most significant differing bit (MSDB) of $V_{1..7}$ and $W_{1..7}$
- [Reminder] CLWW scheme: fix a PRF, F.

mask (pad)

$$\text{PracticalORE}(k_L, V_{1..7}) = p_1 \| \ldots \| p_7 \quad \text{where}$$

$$p_j = \boxed{F(k_L, V_{1..(j-1)})} + v_j \pmod 3$$

  - Each bit is masked by an element of Z_3 derived from the prefix preceding the bit
- Baffle variant, PracticalORE2: essentially the same, but mod 2 instead of mod 3
  - Will reveal location of MSDB($V_{1..7}$, $X_{1..7}$) but not its value.
  - In the scheme, also have Server 1 reveal all of $V_{1..7}$ to the Database so it can uncover the MSDB values.

Baffle Encrypted Comparison

$c_1 = F(k, n_1) - d_1$
$c_2 = F(k, n_2) - d_2$

LESSTHAN? (n_1, c_1, m_1)
(n_2, c_2, m_2)

Database

- Compute $z_0 = x_0 \oplus v_0$
- If $W == Y$, let $\mathcal{v} = v_0$; otherwise, let $\mathcal{v} = v_j$ corresponding to the MSDB between W and Y.
- Return TRUE iff
  $z_0 \oplus \mathcal{v} == 1$

$n_1, c_1, m_1,$
n_2, c_2, m_2

W, V

n_1, n_2

Y, x_0

Server 1

- Check MACs: verify
  $m_1 == M(k_a, n_1, c_1), m_2 == M(k_a, n_2, c_2)$
- Compute ciphertext difference
  $V = c_1 - c_2$
- Set
  $W = \text{PracticalORE2}(k_L, V_{1..7})$

Server 2

- Compute pad difference
  $X = F(k, n_1) - F(k, n_2)$
- Set
  $Y = \text{PracticalORE2}(k_L, X_{1..7})$

Auth. key k_a

ORE key k_L
(ephemeral)

Encryption key k

Implementation Particulars

- Use AES to generate the "pseudorandom" bits in PracticalORE2.
- For each prefix-derived mask, the number of AES output bits needed is 1 + [prefix length]
- Mask = XOR of all AES bits corresponding to 1's in the prefix, XORed with the one extra bit.
  - Extra bit guarantees ≥1 pseudorandom bit used in each mask (even all-0 prefix)
  - Usage of other bits guarantees that different prefixes' masks are independent
- Example:

prefix	`01101`
pseudorandom bits	`101011`
mask	`XOR(0,1,1,1) = 1`

- Relatively efficient: requires only 3 AES blocks to ORE-encrypt a 32-bit character

Security Considerations

- Recall: security model assumes an intruder can only compromise one server at a time

- Adversary at Server 1?
  - All cipher data (derived from plaintext) is masked by a pseudorandom quantity generated using a key unknown at the server.
- Adversary at Server 2?
  - No cipher data.
- Adversary at Database?
  - The interesting case.

Security Considerations: Adversary at Database

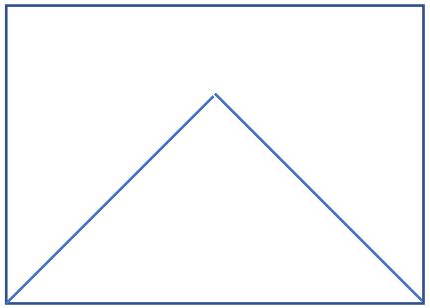
- All bits of V are leaked, but this isn't a big deal (Database could compute $V = c_1 - c_2$ itself)
- Use of PracticalORE2 leaks W only up to its MSDB with V… say, j bits
- Does this mean that only the first j bits are revealed of $d_1 - d_2 = X - V$?
  - No.
  - Consider V as uniformly random over Z_{256}, and X can be thought of as $V + d_1 - d_2$.
  - The probability that the MSDB of $V_{1..7}$ and $X_{1..7}$ is bit $j \in \{0,\ldots,7\}$ is $(d_1 - d_2)/2^{7-j}$. See table.

Example pairs with differing most-significant bit	Value of $d_1 - d_2 = X - V$	Probability that V and X differ in the most-significant ($j = 0$) bit
(A) d_1 = 1000000, d_2 = 0000000	2^6	$2^6/128 = 1/2$
(B) d_1 = 1000000, d_2 = 0111111	1	1/128

Thus, if we see V and X differ in the most-significant bit, case (A) is much likelier than case (B).

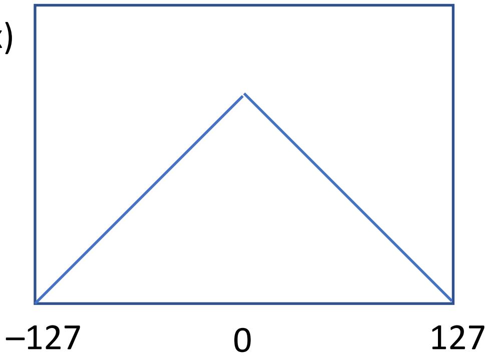
Security Considerations: Adversary at Database

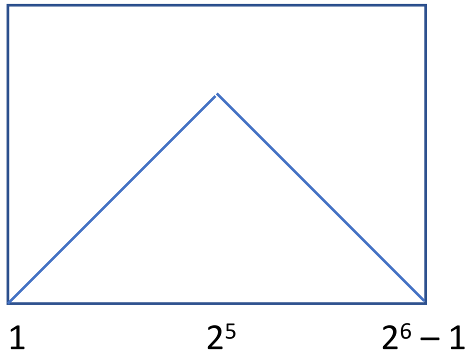
- Theorem. The scheme is semantically secure with leakage function giving the plaintext difference $d_1 - d_2$ between each pair queried.
  - Note this baseline security would be achievable in much simpler & efficient ways
- In practice, more is protected—namely, the difference is only leaked up to a distribution. E.g., if MSDB of X and V is bit $2 \in \{0..7\}$, and it's revealed that $X > V$, then $d_1 - d_2$ is known to follow this distribution:



Baffle Comparison Leakage in Context

- Baffle originally wanted to try to prove that either (a) the MSDBs of d_1 and d_2 or (b) the MSB of $d_1 - d_2$ is leaked, and nothing else.
  - Unfortunately, both are false. (See previous example.)
- But, arguably, these leakage notions are artificial, anyway—they depend on data encoding!
- In fact… what would (a) leaking the MSDBs of d_1 and d_2 tell us about $d_1 - d_2$, anyway?
  - Pretend we don't know anything about common ASCII usage, i.e. we have no a priori knowledge about d_1 and d_2. Then they're uniformly random over Z_{128}. We start with a distribution of $d_1 - d_2$ in the left picture.
  - Revealing the MSDB of d_1 and d_2 improves our knowledge of $d_1 - d_2$. E.g., if they first differ in bit 2 ∈ {0..7}, and $d_1 > d_2$, we have the right picture. (Look familiar?)





Baffle Comparison Leakage in Context

- Observation (informal): effectively, the actual Baffle ($d_1 - d_2$)-leakage of is similar to the desired ($d_1 - d_2$)- leakage of (a) revealing only the MSDBs of d_1 and d_2, if we had no a priori knowledge about d_1 and d_2.

- However, one major difference: MSDB of d_1 and d_2 is deterministic, while Baffle ($d_1 - d_2$)-leakage is randomized based on the computed pads

- Could this similarity be formalized?

Conclusion

- An interesting use case of searchable encryption
- Practical ORE used for an unforeseen application—essentially, on “secret share differences” rather than plaintexts
- Comparison protocol is semantically secure under leakage function giving the difference between queried plaintexts (proved, weak result)
- In fact, less is leaked, but the adversary’s knowledge follows a non-uniform distribution that is not easily captured by a crypto notion.
- The leakage profile doesn’t directly translate to MSDB of plaintexts or MSB of plaintext difference, but there are some interesting similarities between the distribution leaked and the former.

Questions / Comments?

- Thanks for listening.