Generalized Non-interactive Oblivious Transfer Using Count-limited Objects With Applications To Secure Mobile Agents

Vandana Gunupudi
University of North Texas

Stephen R. Tate
University of North Carolina at Greensboro

Overview

- Motivation: Mobile agents
- Oblivious Transfer (Interactive and non-interactive)
- Trusted Platform Modules and clobs
- Generalized non-interactive OT (GNIOT)
  - Problem and solution
  - Theorems and proofs
- GTX protocol
- Some Experimental Results

Motivation: Mobile Agents

- Code and data that migrates within a network and performs autonomous execution at each host
  - Typical agent example: comparison-shopping agent
    - can carry sensitive information like credit card numbers
  - Typically, agent owner (originator) encapsulates agent with required data and functionality
  - Mobile agent performs computations at each host and returns to originator
- Security issues:
  - Protecting host from malicious agents
  - Protecting agent from malicious hosts
    - Various solutions based on Secure Function Evaluation (SFE)

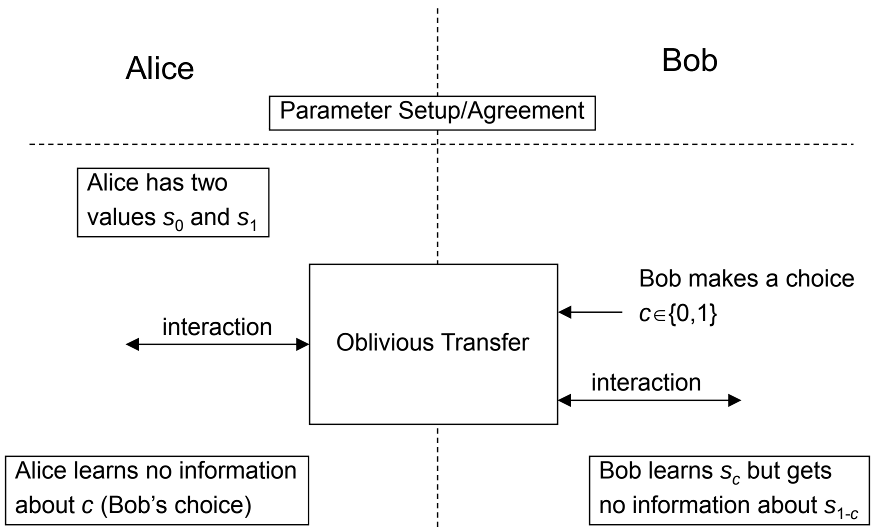
Financial Crypto 2008

2-party Secure Function Evaluation [Yao 1986]

- Two parties evaluate a function such that each party behaves honestly and learns nothing more than it is entitled to.
  - Inputs: Alice holds value a
Bob holds value b
  - Computation: Compute f(a,b) $\rightarrow$ (A,B)
  - Output: Alice gets A
Bob gets B
- Security:
  - Alice learns no more about B than follows from a and A
  - Bob learns no more about A than follows from b and B
- How does Bob get his input?
  - Bob gets encrypted input bit-by-bit from Alice by using 1-out-of-2 OT

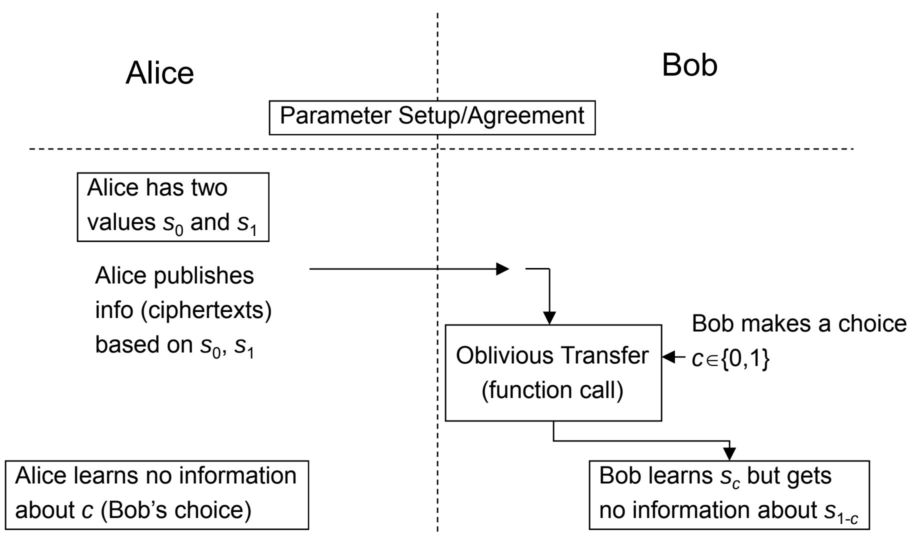
Financial Crypto 2008

Standard Oblivious Transfer



Financial Crypto 2008

Non-Interactive Oblivious Transfer



Financial Crypto 2008

Impossibility in the Standard Model

- Once Bob receives Alice's published values, takes a "snapshot" of his state
- Next picks c=0 and computes s_0
- Then "rolls back" state to earlier snapshot
- Picks c=1 and computes s_1

Key Point: In the standard model, a party can completely examine and manipulate (restore) it's own state.

Note: An earlier "non-interactive" OT (Bellare and Micali) was very different - Bob didn't get to make a choice and received a randomly selected s_c.

Hardware Extensions to the Rescue!

- "Trusted Computing" initiative
  - Spearheaded by the Trusted Computing Group
  - Hardware (Trusted Platform Modules) becoming more common
- Among other capabilities, a TPM:
  - Manages and controls use of keys
  - Supports a Monotonic Counter
    - After an increment, can never be reset
    - State that can't be restored!

- Note: We don't need other features of TPMs
- Can use smart-cards or any crypto processors that control key usage

Virtual Monotonic Counters (Sarmenta et al. 2006)

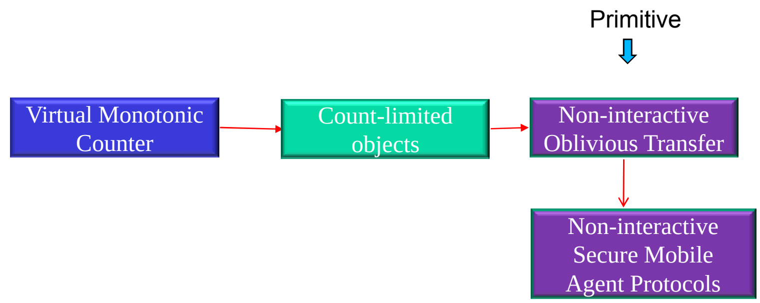
- Large number of counters that can be:
  - Initialized
  - Incremented
  - Cannot be reset to any previous value
- Count –Limited Objects (Keys)
  - Objects that can only be used a limited number of times
  - Each clob linked to a *dedicated* virtual monotonic counter to track usage of the clob
  - Examples: n-time-use delegated signing/encryption keys
- Our applications of clobs
  - *Non-interactive form of Oblivious Transfer*

Non-interactive OT (with clobs)

- Obvious use for 1-out-of-2 OT:
  - Bob (with access to a TPM) generates a 1-time use keypair (K_p,K_s)
  - Sends K_p to Alice with certificate
  - Alice verifies clob and encrypts both values with K_p
  - Bob can decrypt only 1 value (TPM enforces this)
- Problem:
  - Many applications (e.g., SFE) require multiple OTs
  - We need a separate clob for each value, and multiple key generations (expensive!)
- Our solution: Uses a *single* clob for multiple, general OTs

Our Contributions

- Definition of "Generalized Non-interactive Oblivious Transfer"
- An efficient implementation of GNIOT for TPM-enhanced models
- Careful security analysis and rigorous proofs of our implementation
- Use of the GNIOT primitive to create a new non-interactive, secure agent protocol



Financial Crypto 2008

Generalized Non-interactive OT

- **Setup Phase**: K_p and K_s public/secret key info $(\mathcal{K}_p, \mathcal{K}_s) \leftarrow Setup(1^\lambda)$
- **Transmit phase**: n independent $k_i - out - of - m_i$ OTs

$$x_{i,j} \quad i \in \{1, 2, \cdots, n\} \text{ and } j \in \{1, 2, \cdots, m_i\}$$

$$C \leftarrow Transmit_{\mathcal{K}_p} \begin{pmatrix} \langle k_1, x_{1,1}, x_{1,2}, \cdots, x_{1,m_1} \rangle, \\ \langle k_2, x_{2,1}, x_{2,2}, \cdots, x_{2,m_2} \rangle, \\ \vdots \\ \langle k_n, x_{n,1}, x_{n,2}, \cdots, x_{n,m_n} \rangle \end{pmatrix}.$$

- **Decrypt Phase**

$$(t_k, \mathcal{S}_k) \leftarrow Decrypt_{\mathcal{K}_s}(\mathcal{S}_{k-1}, C, i_k, j_k) \quad \text{for } k = 1, 2, \ldots, q \text{ for some number of queries } q$$

$$(i_k, j_k) \leftarrow ind(t_k)$$

- **Post Process phase**:

$$\langle v_1, v_2, \ldots, v_q \rangle \leftarrow PostProcess(t_1, t_2, \cdots, t_q)$$

Financial Crypto 2008

Our TPM-based scheme

Setup Phase.: Bob creates an N-time use count limited key pair (K_p, K_s), where $N = (k_1 + k_2 + \cdots + k_n)$.

Transmit Phase: $R = R_1 \oplus R_2 \oplus \cdots \oplus R_n$ each i we compute m_i shares of each R_i denote the shares of R_i by $f_i(j)$, for $j = 1, \ldots, m_i$

$$C_{i,j} = \mathcal{PKE}_{K_p}(\langle \mathcal{SKE}_R(x_{i,j}), f_i(j) \rangle).$$

Decrypt Phase: $Decrypt_{\mathcal{K}_s}(\mathcal{S}, C, i_k, j_k)$ then just uses $\mathcal{K}_s$ to decrypt C_{i_k, j_k},

$$t_k = \langle i_k, j_k, \mathcal{SKE}_R(x_{i_k, j_k}), f_{i_k}(j_k) \rangle .$$

- PostProcess: Reconstruct R and decrypt t_k values
- Index set: set of indices (i,j) $I(i) = \{ j \mid (i, j) \in I \}$
- Well formed index set: $| I(i) | = k_i \forall i \in \{1, \cdots n\}$

Financial Crypto 2008

GNIOT Game

Adversary A supplies plaintext input where each input has 2 possibilities: $x^0_{i,j}$, $x^1_{i,j}$ for $i=1,2,\ldots n$ and $j=1,2,\ldots m_i$

Oracle generates an independent random bit $r_{i,j} \in_R \{0,1\}$ for each pair.

Oracle creates a single input X using $x_{i,j}^{r_{i,j}}$ and calls the Transmit function which returns C.

A makes a series of calls to the Decrypt function which returns $t_1, t_2, \ldots, t_q$.

A is free to make calls to the PostProcess function.

Finally, A outputs a guess g and an index (a,b).

A wins the game if g = $r_{a,b}$. Formally,

$$Adv_{GNIOT,\mathcal{A}} = \left| Pr[g = r_{a,b} | (a, b) \notin \mathcal{I} \text{ or } \mathcal{I} \text{ not well-formed}] - \frac{1}{2} \right| .$$

Financial Crypto 2008

Security Analysis

Theorem 5.3. *If PKE is an IND-CCA2 secure public key scheme and SKE is a IND-CCA2 secure symmetric cipher, then the GNIOT game can be won by a probabilistic, polynomial time adversary $\mathcal{A}$ if and only if $\mathcal{I}$ is a well formed index set and $(a, b) \in \mathcal{I}$.*

- Similar to "hybrid encryption" (Public key + symmetric cipher)
  - Hybrid encryption proofs due to [Cramer and Shoup, 1998]
  - Proof: Composition of secure components is secure
  - Proof is broken into 3 cases

Financial Crypto 2008

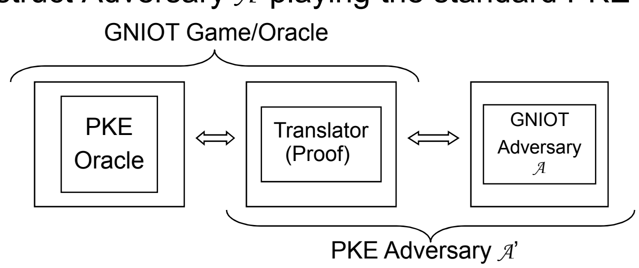
Proof

Case 0 $(a, b) \in \mathcal{I}$, and $\mathcal{I}$ is a well-formed index set.

- If you follow the rules, you win the game

Case 1 $(a, b) \notin \mathcal{I}$, where $\mathcal{I}$ is a well-formed index set.

- Adversary $\mathcal{A}$: PPT machine playing GNIOT game
- Construct Adversary $\mathcal{A}'$ playing the standard PKE game



Financial Crypto 2008

Proof Sketch for Case 1

- *Basic Idea*: Treat as multiple PKE (CCA2) games, and guess which one really "counts"
- Step 1 (setup): Get public key from PKE oracle and generate R (and shares)
- Step 2 (send): $\mathcal{A}$ passes to $\mathcal{A}'$: $x^0_{i,j}$, $x^1_{i,j}$ for i =1,2,…,n and j =1,2,…m_i
- Step 3: $\mathcal{A}'$ creates C for $\mathcal{A}$?: Pick an index (a,b) at random
  - For all $(i,j) \neq (a,b)$:
    - Pick random r_{ij} and compute PKE.Encrypt(SKE$_R$(x_{ij}^{rij}),$f_i(j)$)) [this is c_{ij}]
  - For index (a,b):
    - Submit (SKE$_R$($x^0_{a,b}$),$f_i(j)$)) and (SKE$_R$($x^1_{a,b}$),$f_i(j)$)) to PKE oracle which returns encryption of one of these values [this is c_{ab}].
  - C is collection of all c_{ij}'s

Proof Sketch for Case 1 – cont'd

- How does $\mathcal{A}'$ handle decryption requests from $\mathcal{A}$?
  - If $(i,j) \neq (a,b)$, then $\mathcal{A}'$ processes decryption query correctly
  - Else: $\mathcal{A}'$ loses the game
- Finally, $\mathcal{A}$ outputs (a',b') and guess g
  - If $(a',b') \neq (a,b)$, then $\mathcal{A}'$ loses PKE game
  - Else $\mathcal{A}'$ outputs g as its guess in the PKE game

Proof Sketch continued

Probability bounds for A winning the GNIOT game:

- $\mathcal{A}'$ wins the game if and only if
  - $(a,b) = (a',b')$ [which occurs with probability $1/N$], and
  - $\mathcal{A}$ wins the GNIOT game

So:

$$\Pr[\mathcal{A}' \text{ wins}] = (1/N) \cdot \Pr[\mathcal{A} \text{ wins}]$$

$$\Pr[\mathcal{A} \text{ wins}] = N \cdot \Pr[\mathcal{A}' \text{ wins}] \leq N \cdot Adv_{PKE}$$

Since Adv_{PKE} is negligible, probability that $\mathcal{A}$ wins GNIOT is negligible.

Proof Sketch, continued

Case 2: $(a,b) \in \mathcal{I}$, but $\mathcal{I}$ is not a well-formed index set

Bottom line:

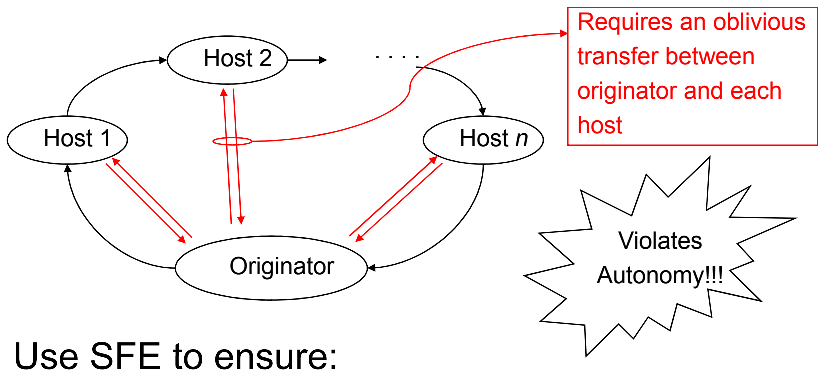
$$\Pr[\mathcal{A} \text{ wins}] \leq 2\, Adv_{SKE} + Adv_{PKE}$$

Intuition: $\mathcal{A}$ must either

- Break PKE to get additional shares of R, or
- Break SKE to get plaintext without reconstructing R

Details: See the paper

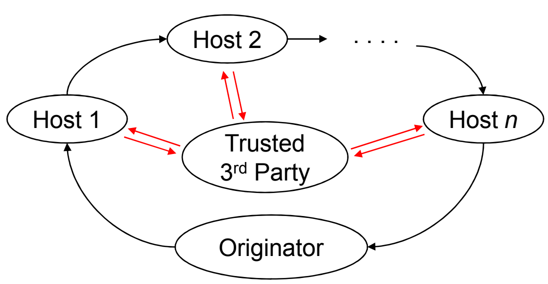
Oblivious Transfer and Agents



- Use SFE to ensure:
  - Confidentiality and integrity of agent state
  - As much confidentiality as possible for host input

Financial Crypto 2008

Software-only solution



- Due to [Algesheimer,Cachin,Camenisch,Karjoth, 2001]
- Trusted 3rd party acts as “stand-in” for originator in OT
  - TTP must not reveal host inputs to originator
  - TTP must not allow hosts to access agent state *or run multiple trials*

Financial Crypto 2008

Mobile Agent Security Issues

- Software-only solutions for protecting privacy of agent data
  - ACCK Protocol: Uses a trusted third party (TTP)
    - Joy Algesheimer, Christian Cachin, Jan Camenisch, and Gunter Karjoth, "Cryptographic security for mobile code," in *Proc. IEEE Symposium on Security and Privacy*, May 2001, pp. 2-11.
  - TX Protocol: Uses threshold cryptography and multiple agents to obviate need for TTP
    - Stephen R. Tate and Ke Xu, "Mobile Agent Security Through Multi-Agent Cryptographic Protocols", in *Proc. of the 4th International Conference on Internet Computing (IC 2003),* pages 462-468.
- Hardware-assisted solution
  - GTX protocol uses GNIOT primitive

Overview of GTX Protocol

- All hosts have TPMs and execute Setup phase of GNIOT prior to start of protocol
- Originator:
  - Executes Transmit phase for each host input bit (n-bits)
  - Adds output of GNIOT Transmit phase to agent
- Host:
  - Calls GNIOT Decrypt on the correct index set
  - Calls GNIOT PostProcess with output of GNIOT Decrypt to obtain exactly the correct number of inputs required
- Non-interaction property:
  - The host and originator need not contact each other after the Transmit phase
- All other protocols require some form of interaction when the agent reaches the host

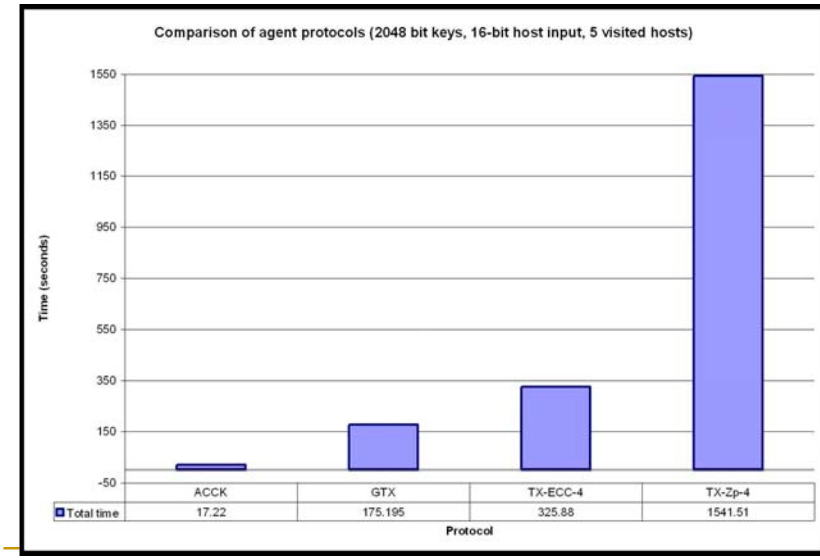
Practical aspects

- Experimental results with GTX protocol
  - TPM Simulator
- SAgent framework: platform for testing GTX protocol
- Comparison of GTX to other secure agent protocols

SAgent

- Security framework we designed for the JADE platform
- Designed for comprehensive protection of mobile agent data
- Secure agent protocols very complex
- Purpose of SAgent: design a simple, usable interface that abstracts protocol details
- Abstracted interface handles various secure agent protocols
- GTX added to SAgent

GTX Performance Analysis



Financial Crypto 2008

Conclusion

- Showed how to remove interaction requirements in OT
- Provide rigorous security proofs for our GNIOT construction
- Apply GNIOT primitive to secure agent computations
- Showed GTX protocol is efficient

Financial Crypto 2008