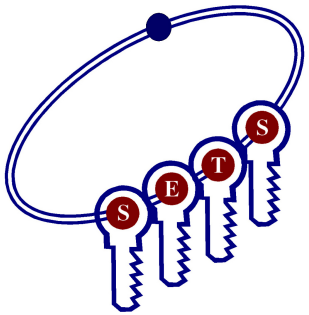
Revisiting Single-server Algorithms for Outsourcing Modular Exponentiation

Jothi Rangasamy Lakshmi Kuppusamy

Society for Electronic Transactions and Security [SETS]
Chennai, India



Strategy and Synergy for Security

INDOCRYPT 2018

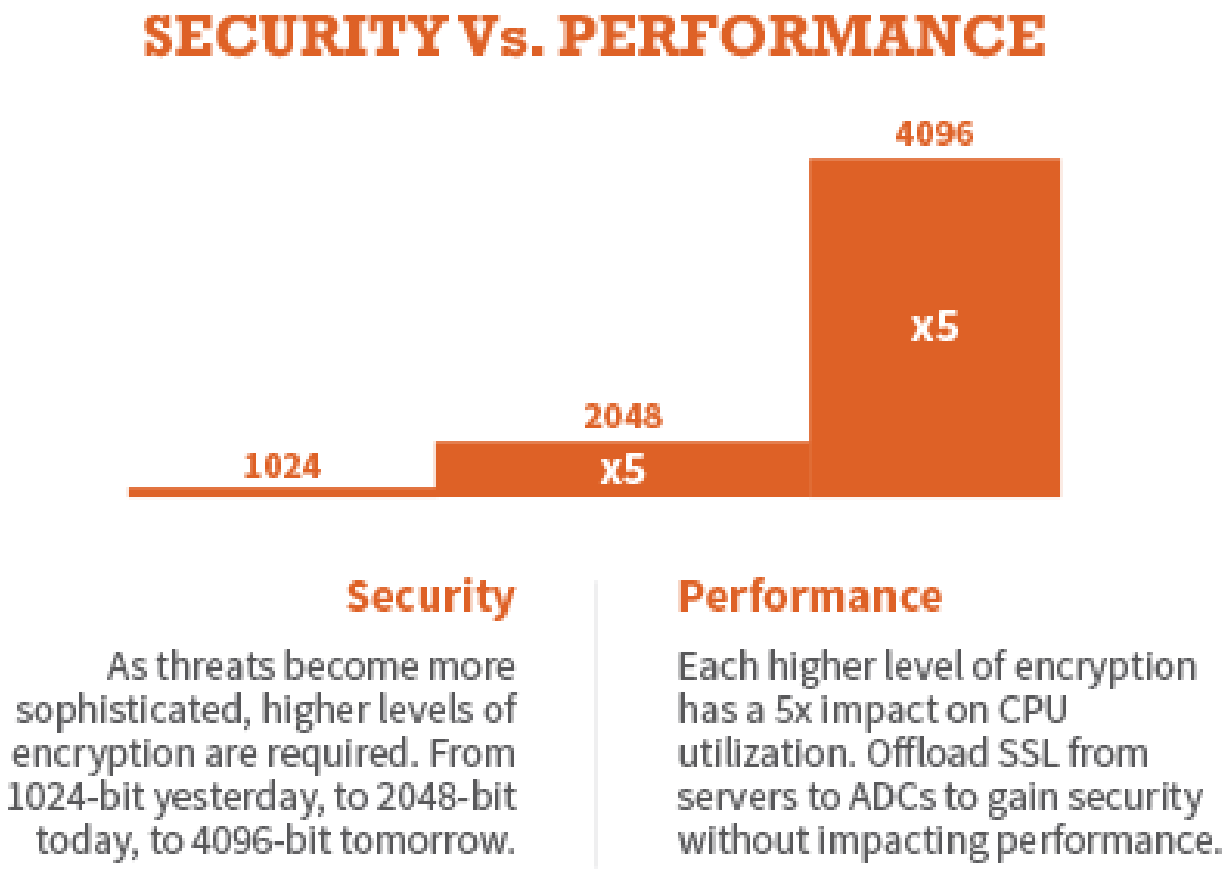
Special thanks to Dr Uzunkol (University of Hagen, Germany) for providing his material and suggestions.

Outline

Basic Computations in PKC

- Public-Key Cryptography helps us address many security concerns.
  - Digital signatures for authentication
  - Key establishment protocols for computing session keys.
- Operations such as modular exponentiation and bilinear pairing are predominantly used in PKC.
- *IoT and Handheld* devices are going to dominate, and often, cannot afford performing complex cryptographic operations.
- A Solution: Offload intensive operations.

Busy Servers also need...



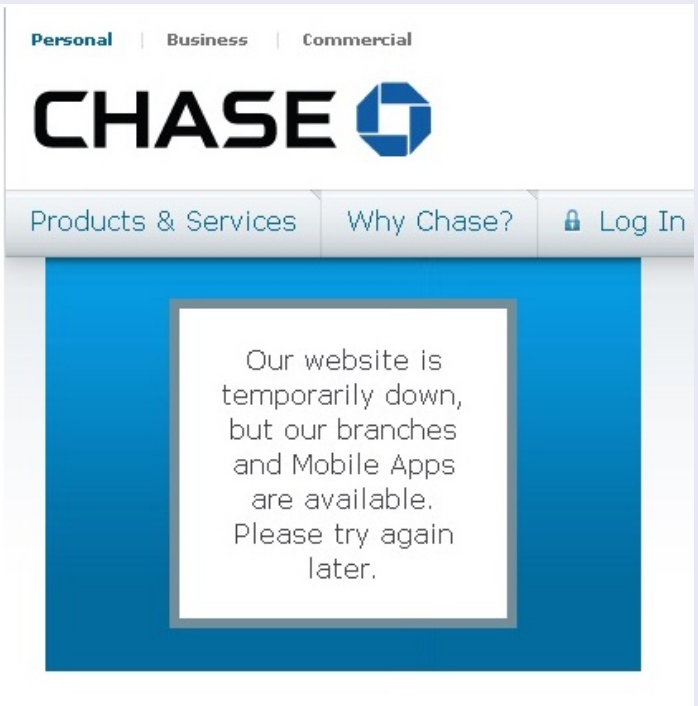
(Source: Google.com)

- Denial-of-Service (DoS) attacks aim to make the targeted service unavailable or slow down by exhausting server resources.

An Example

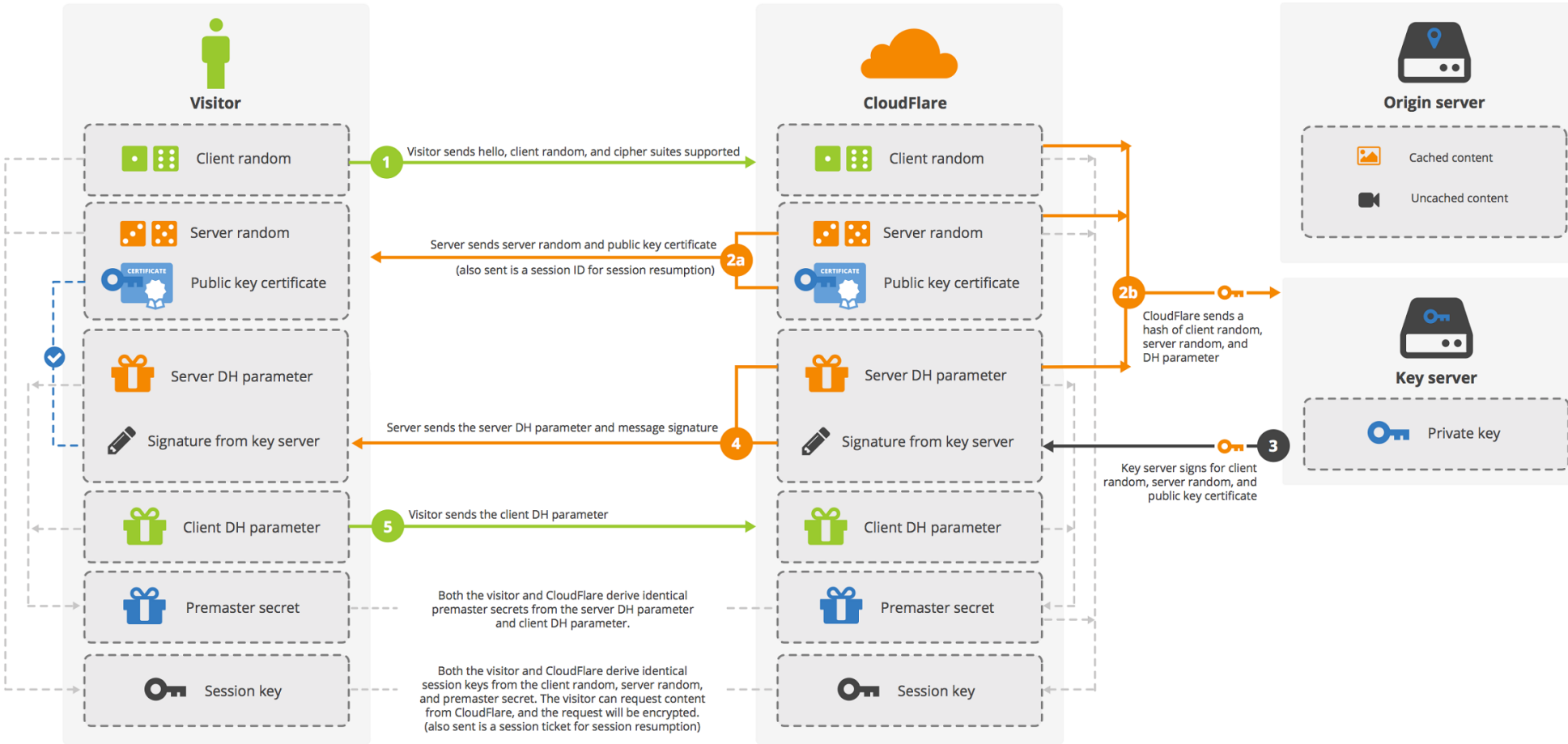
Major US banks and FIs (2013)

BANK OF AMERICA, GOLDMAN SACHS, CITIGROUP, and CHASE websites were taken down for several hours, blocking customers from their accounts and otherwise disrupting business.



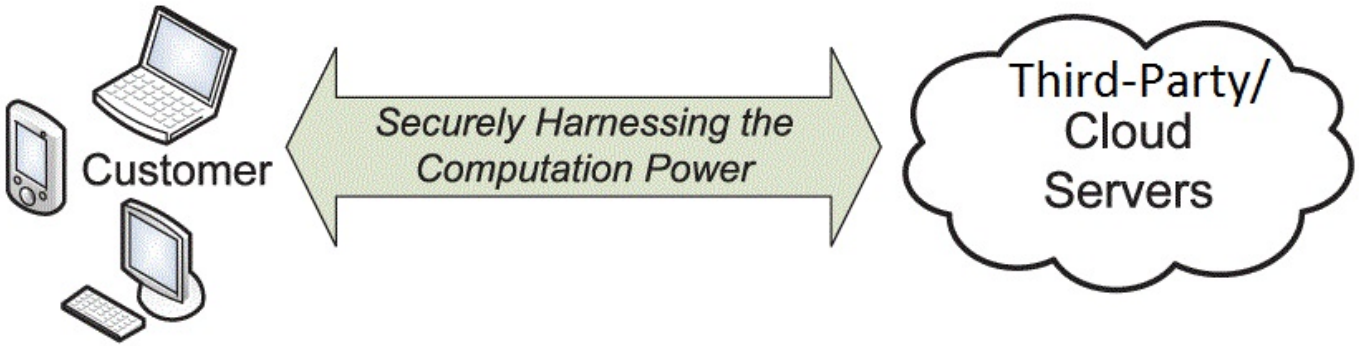
(Source: www.cnet.com)

SSL is offloaded to a trusted cloud server.



(Source: www.cloudflare.com)

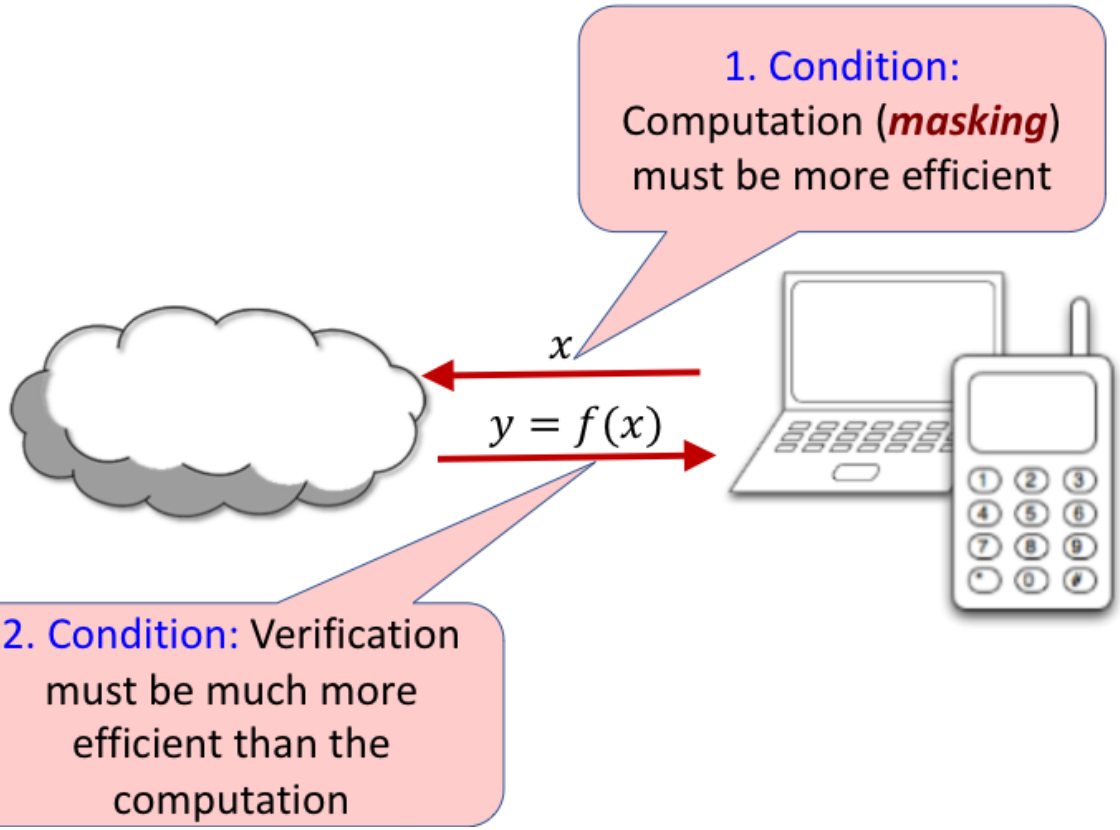
Outsourcing $\mathbf{u}^{\mathbf{a}} \bmod \mathbf{p}$



Security issues

- Does delegation reveal (confidential or sensitive) data to computing service provider?
- Is computation performed correctly without errors?
- $\Rightarrow$ CSP: Is it benign, semi-honest, or even malicious?

Efficiency Requirements



Expectation

Randomizing + Communication + Derandomizing + Verification << Local Computation

Framework of Hohenberger and Lysyanskaya (TCC 2005)

- Introduced the following security notions
  - **Secrecy**: A (possibly malicious) server should not learn any information about the values given in masked form.
  - **Checkability**: The device which outsources the task should be able to detect a cheating server, except with negligible probability.
- Proposed the first secure outsourcing algorithm for variable-base, variable-exponent modular exponentiation with checkability 1/2.

Further Work

Followed by several notable results in terms of improving efficiency and error detecting probability: Chen *et al* (ESORICS 2012), Wang *et al* (ESORICS 2014) and Kiraz-Uzunkol (2016).

Overall idea

Delegation of $u^a \in \mathbb{G} =< g >(\subset \mathbb{Z}_p^*)$

1. Precomputation: Normally involves invocation of the subroutine Rand) to generating pairs of the form $(t, g^t) \in \mathbb{Z}_q^* \times \mathbb{G}$
2. Randomization of both a and u with Rand
3. Delegation to the server(s)
4. Verification and derandomization of the output
5. Computing the actual output (only group multiplication!)

Cai *et al* Scheme (2017)

Precomputation

- $g^{\beta}, g^{\gamma}, g^{\theta} \leftarrow \text{RAND}$

Masking Inputs

- $u^a = g^{\beta} g^{\gamma} w^a$
- $u^a = g^{\theta} g^{\tau} v^a$
- $a_1 = a - 2^i$
- $a_2 = a - 2^j$

Recovery

- $u^a := B(\delta g^{\gamma}) w^{a_1} m[i]$

Queries and Verification

- $(g^{t_1}, \gamma/t_1, p) \rightarrow \delta g^{\gamma}$;
- $(wg^{s_1}, a_1, p) \rightarrow R_{11} = \delta w^{a_1} g^{s_1 a_1}$;
- $(g^{s_3}, \frac{s_1 a_1 - s_2}{s_3}, p) \rightarrow R_{12} = \delta g^{s_1 a_1 - s_2}$;
- $(g^{t_2}, \tau/t_2, p) \rightarrow \delta g^{\tau}$;
- $(vg^{s_4}, a_2, p) \rightarrow R_{21} = \delta v^{a_2} g^{s_4 a_2}$;
- $(g^{s_6}, \frac{s_4 a_2 - s_5}{s_6}, p) \rightarrow R_{22} = \delta g^{s_4 a_2 - s_5}$;
- $(m_1, 2) \rightarrow m[1] = m_1^2$;
- $(m_2, 2^2) \rightarrow m[2] = m_2^4$;
- $\cdots$
- $(w, 2^i) \rightarrow m[i] = w^{2^i}$;
- $\cdots$
- $(v^{-1}, 2^j) \rightarrow m[j] = v^{-2^j}$;
- $\cdots$
- $(m_n, 2^n) \rightarrow m[n] = m_n^{2^n}$;

After receiving the outputs, $\mathcal{C}$ checks

$$B\delta g^{\gamma} w^{a_1} m[i] m[j] \stackrel{?}{=} D\delta g^{\tau} v^{a_2} \mod p.$$

Similar attack on Li et al (2016)

Zhou et al Scheme (ExpSOS)

Aim: compute $u^a \bmod N$ where $N \leftarrow$ prime/RSA modulus.

Masking Inputs u, a, N :

- $L \leftarrow pN$
- $U \leftarrow u + rN \bmod L$
- $A_1 \leftarrow a + k_1\phi(N)$,
- $A_2 \leftarrow t_1 a + t_2 + k_2\phi(N)$, $t_1, t_2 \leq b$

Queries to $\mathscr{U}$ and Verification

$\mathscr{C}$: $R_1^{t_1} \cdot u^{t_2} \stackrel{?}{\equiv} R_2 \mod N$

$\xrightarrow{(U, A_1, A_2, L)}$ $\mathscr{U}$: $R_1 = U^{A_1} \mod L$, $R_2 = U^{A_2} \mod L$

$\xleftarrow{R_1, R_2}$

Recovering u^a $\quad u^a \equiv R_1 \mod N$.

Attacking Secrecy of ExpSOS

Secret exponent a can be recovered in polynomial time if reused atleast once in another round.

- Assume the client wants to compute $u^a \mod N$ in first run and $(u')^a \mod N$ in second run.

  Masking a in first run

  - $A_1 \leftarrow a + k_1\phi(N)$,
  - $A_2 \leftarrow t_1 a + t_2 + k_2\phi(N)$, $t_1, t_2 \leq b$

  Masking a in next run

  - $A'_1 \leftarrow a + k_3\phi(N)$,
  - $A'_2 \leftarrow t_3 a + t_4 + k_4\phi(N)$, $t_3, t_4 \leq b$

  Recovering a

  $A_1 - A'_1 = (k_1 - k_3)\phi(N)$.

  Given multiples of $\phi(N)$, the secret exponent a can be found in polynomial time using the Miller's algorithm or with GCD algorithm.

MExpSOS : Our Algorithm for Single Exponentiation u^a

Aim: compute $u^a \bmod N$ where $N \leftarrow$ prime/RSA modulus.

Masking Inputs u, a, N :

- $L \leftarrow pN$
- $U \leftarrow u + N' \mod L$ where $N' \leftarrow rN(fixed)$
- $a_1 \leftarrow a - t_1$
- $A_1 \leftarrow a_1 + k_1\phi(N)$
- $A_2 \leftarrow t_2 a + k_2\phi(N)$, $t_1, t_2 \leq b$

Queries to $\mathscr{U}$ and Verification

$\mathscr{C}$		$\mathscr{U}$
	$\xrightarrow{(U, A_1, A_2, L)}$	$R_1 = U^{A_1} \mod L$
$(R_1 u^{t_1})^{t_2} \mod N \stackrel{?}{=} R_2 \mod N$	$\xleftarrow{R_1, R_2}$	$R_2 = U^{A_2} \mod L$

Recovering u^a $\quad u^a \equiv R_1 u^{t_1} \mod N$.

πExpSOS : New Algorithm for Simultaneous case $\prod_{i=1}^{n} u_i^{a_i} \bmod p$

Aim: compute $\prod_{i=1}^{n} u_i^{a_i} \bmod N$ where $N \leftarrow$ prime/RSA modulus.

Masking Inputs u_i, a_i, N :

- $L \leftarrow pN$
- $U_i \leftarrow u_i + N' \mod L$ where $N' \leftarrow rN(fixed)$
- $a_{1i} \leftarrow a_i - t_1$
- $A_{1i} \leftarrow a_{1i} + k_{1i}\phi(N)$
- $A_{2i} \leftarrow t_2 a_i + k_{2i}\phi(N)$

$2n$ Queries to $\mathscr{U}$ and Verification

$$\left[\prod R_{1i}(\prod u_i)^{t_1}\right]^{t_2} \stackrel{?}{\equiv} \prod R_{2i} \mod N$$

$\mathscr{C}$ sends (U_i, A_{1i}, A_{2i}, L) to $\mathscr{U}$; $\mathscr{U}$ returns R_{1i}, R_{2i}.

$$R_{1i} = U^{A_{1i}} \mod L$$
$$R_{2i} = U^{A_{2i}} \mod L$$

Recovering $u_i^{a_i}$ $\prod_{i=1}^{n} u_i^{a_i} \equiv \prod_{i=1}^{n} R_{1i}(\prod_{i=1}^{n} u_i)^{t_1} \mod N$.

Comparison with Known Algorithms

Algorithm	Secrecy	verifibility Probability	Pre-processing Required
Cai *et al.*	Yes	0	Yes
Li *et al.*	Yes	0	Yes
Kiraz-Uzunkol	Yes	< 1	Yes
ExpSOS	No	≈ 1	No
MExpSOS, πExpSOS	Yes	≈ 1	No

Table: Single server-based outsourcing algorithms and their properties

Security and Verifiability

- *Secrecy* for the exponent a is proven under Hohenberger-Lysyanskaya framework using One Malicious Server model
- Security for the base u and *verifiability* hold if the factorization of n is intractable
- Verifiability is $1 - \varepsilon$, (ε is negligibly small) for any polynomial-time adversary.

Conclusion and Future Work

- Found security issues with recent algorithms.
- Proposed algorithms for single and simultaneous cases do not require precomputation
- More efficient and hence more practical.
- To design a secure outsourcing algorithm to a single (untrusted) server without small exponentiation in verification.

Revisiting Single-server Algorithms for Outsourcing Modular Exponentiation

THANK YOU!

jothi.rangasamy@gmail.com