Verifying Location-Based Services with Declassification Enforcement*

Cong Sun, Sheng Gao, and Jianfeng Ma

School of Computer Science and Technology, Xidian University, China
suncong@xidian.edu.cn

Abstract. Location privacy has been considered to be an important issue along with the advances of location technologies and the pervasive use of location based services. Although a variety of location privacy techniques have been developed, language-based techniques have rarely been used on privacy enforcement of location-based services. In this work, we propose a verification framework to enforce the privacy preservation of locations. The enforcement leverages reachability analysis of pushdown system to ensure that the service-specific data aggregation functions comply with the privacy property. The approach supports inter-procedural analysis and is more precise than existing work.

1 Introduction

Location-based services (LBS) are a category of services that integrate the location information of mobile nodes with other information to provide easily accessible added value to mobile users. LBS usually require users' current location information to answer their location-based queries. Along with the widely use of LBS, user that employs LBS frequently faces potential privacy problem. The privacy of user is threatened as LBS require the user to release her private location information to them and these data allow for profiling the user's movement, health condition, or affiliation. Therefore Developing enforcement mechanisms of privacy requirements to preserve the private location information for mobile users becomes an emergency. There have been several paradigms of architectures for privacy-preserving LBS [3,12] where the trade-offs between anonymity of private location information and the quality of service are discussed. In the *trusted third party architecture*, a static analyzer can be developed on the trusted server to ensure the data aggregation function of private location information can meet with certain privacy requirement. By verifying the possibly malicious aggregation function code from the untrusted LBS provider, this approach can provide a service-specific privacy without loss of quality of service.

In this work we propose a verification framework for the trusted server to enforce privacy property that prevent unintentional leak of users' private location

* This work was supported by the Key Program of NSFC-Guangdong Union Foundation (U1135002), Major national S&T program(2011ZX03005-002), National Natural Science Foundation of China(60872041,61072066), the Fundamental Research Funds for the Central Universities(JY10000903001JY10000901034), and GAD Advanced Research Foundation (9140A15040210HK6101).

Q.Z. Sheng et al. (Eds.): APWeb 2012, LNCS 7235, pp. 695–702, 2012.

information. The typical application scenario is given in Fig. 1. It involves three kinds of parties: 1) the trusted server maintaining all the location information of mobile nodes, 2) the location information provider on each mobile node which sends private location information to the trusted servers, 3) the LBS which defines service-specific data aggregation function with a piece of code, sends the code to the trusted servers for computation and receives the results of computation from the trusted servers. The aggregation function developers use explicit outputs to claim the location information the LBS requires. But without careful examination, the malicious or cursorily developed aggregation function may leak the private location information in a unintentional way. Therefore a verification framework is needed by the trusted servers to ensure this unintentional leakage cannot happen.

The trusted servers first use symbolic pushdown system to model the aggregation functions, then use reachability analysis to decide whether the declassified results of computation meet with the privacy property specified in Section 2.2. If the aggregation function satisfies the privacy property, the trusted server feeds the function with the location information and the data from LBS, and sends the result of computation back to the LBS. We transform the pushdown model of aggregation function using a form of self-composition [1] complemented with a set of auxiliary pushdown rules in order to reduce the declassification-based privacy property to a safety property on the model after transformation. The reachability analysis is then performed on the derived model. A store-match pattern is inherited from our previous work to reduce the state space and to constrain the pre-condition of privacy property.

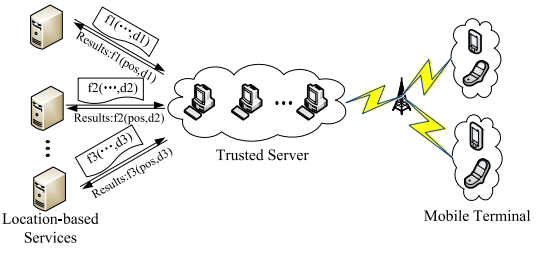

Fig. 1. Application Scenario

Numerous studies have focused on privacy issues in LBS. The pseudonym-based approaches, which prevent any user to communicate with the services [2,5], may greatly influence the QoS. An alternative category of approaches is obfuscation [4,8]. These work mostly focus on the anonymity of locations. On the other hand, our approach mainly focuses on how to validate a privacy-preserved usage, i.e. data aggregation, of users' location information. The most closely related work is that of Ravi et al [6]. They specify the privacy requirement with the *non-inference* property and use a global data-flow analysis to solve whether any explicit flow of location information to output variables exists. Compared with our work, their approach imprecisely reports more false-negatives, and cannot support inter-procedural analysis.

2 Approach

2.1 Reachability Analysis of Pushdown System

We use symbolic pushdown system [9] as the model of aggregation function code. A pushdown system is a stack-based state transition system whose stack

Table 1. Model Construction

c	$\Phi(c, \gamma_j, \gamma_k)$
skip	$\{\langle\gamma_j\rangle \hookrightarrow \langle\gamma_k\rangle \ rt(\mu)\}$
$x := e$	$\{\langle\gamma_j\rangle \hookrightarrow \langle\gamma_k\rangle \ (x' = e) \wedge rt(\mu \setminus \{x\})\}$
if e **then** c_1 **else** c_2	$\{\langle\gamma_j\rangle \hookrightarrow \langle\gamma_t\rangle \ rt(\mu) \wedge e\} \cup \Phi(c_1, \gamma_t, \gamma_k) \cup$
	$\{\langle\gamma_j\rangle \hookrightarrow \langle\gamma_f\rangle \ rt(\mu) \wedge \neg e\} \cup \Phi(c_2, \gamma_f, \gamma_k)$
while e **do** c'	$\{\langle\gamma_j\rangle \hookrightarrow \langle\gamma_t\rangle \ rt(\mu) \wedge e\} \cup \Phi(c', \gamma_t, \gamma_j) \cup \{\langle\gamma_j\rangle \hookrightarrow \langle\gamma_k\rangle \ rt(\mu) \wedge \neg e\}$
$c_1; c_2$	$\Phi(c_1, \gamma_j, \gamma_{\text{mid}}) \cup \Phi(c_2, \gamma_{\text{mid}}, \gamma_k)$
$x := f(y)$	$\{\langle\gamma_j\rangle \hookrightarrow \langle f_{\text{entry}}\gamma_t\rangle \ (z' = y) \wedge rt(\mu_{\text{glb}}) \wedge rt_2(\mu_{\text{loc}})\} \cup$
	$\{\langle f_{\text{exit}}\rangle \hookrightarrow \langle\epsilon\rangle \ (iret' = \ldots) \wedge rt(\mu_{\text{glb}} \setminus \{iret\})\} \cup$
	$\{\langle\gamma_t\rangle \hookrightarrow \langle\gamma_k\rangle \ (x' = iret) \wedge rt(\mu \setminus \{x\})\}, \quad (z \in \mu_{\text{loc}_f})$
$output(x)$	$\{\langle\gamma_j\rangle \hookrightarrow \langle\gamma_k\rangle \ (O'[q] = x) \wedge (q' = q + 1) \wedge rt(\mu \setminus \{O[q], q\})\}$

contained in each state is of unbounded length. A symbolic pushdown system is a compact representation of pushdown system encoding the variables and computations symbolically.

Definition 1 (Symbolic Pushdown System). *Symbolic Pushdown System is a triple* $\mathcal{P} = (\mathcal{G}, \Gamma \times \mathcal{L}, \Delta)$. $\mathcal{G}$ *and* $\mathcal{L}$ *are respectively the domain of global variables and local variables.* Γ *is the stack alphabet.* Δ *is the set of symbolic pushdown rules* $\{\langle\gamma\rangle \hookrightarrow \langle\gamma_1 \cdots \gamma_n\rangle(\mathcal{R}) \mid \gamma, \gamma_1, \cdots, \gamma_n \in \Gamma \wedge \mathcal{R} \subseteq (\mathcal{G} \times \mathcal{L}) \times (\mathcal{G} \times \mathcal{L}^n) \wedge n \leq 2\}$.

The relation $\mathcal{R}$ specifies the environment transformers that direct a single step of symbolic computation according to the pushdown rule. The stack symbols denote the flow graph nodes of program. The model construction function Φ is given in Table 1. It adds constraints to regulate each $\mathcal{R}$ of pushdown rule. The constraint is expressed with logical operation on abstract variables. $\mu = \mu_{\text{glb}} \cup \mu_{\text{loc}_f}$ is a memory store mapping both the global variables and the local variables of current procedure f to values. For the abstraction of output command, we suppose the global linear list O and the index q for next output element to have $\{O, q\} \subseteq dom(\mu_{\text{glb}})$. $iret$ is a global variable temporarily storing the returned value of f. rt means retainment on value of global variables and on value of local variables of the procedure locating the pushdown rule. rt_2 for a rule $\langle\gamma_j\rangle \hookrightarrow \langle f_{\text{entry}}\gamma_k\rangle$ denotes retainment on value of local variables of the caller of procedure f. The back-end algorithm we used for reachability analysis is the symbolic form of saturation algorithm $post^*$ adapted from [9, Sec.3.3.3]. The reachability analysis of certain flow graph node γ actually finds whether the $\mathcal{P}$-automaton $\mathcal{A}_{post^*}$ can accept a configuration whose top stack symbol is γ.

2.2 Specification of Privacy

From a perspective of trusted third party, the LBS should explicitly specify what they want to learn from the private location information, e.g., through the outputs and explicit declassification primitives in their code [6]. The trusted server ensures the explicit outputs of a single run of aggregation function will not release the private location itself. In this work, we use the end-to-end enforcement

of declassification requirement to ensure that the intentional release of information cannot leak any confidential information other than what the LBS claim to obtain through the explicit outputs.

For any aggregation function f, the set of parameters w.r.t. LBS data is $L = \{\ell_1, \ldots, \ell_m\}$ and the set of parameters w.r.t. private location information is $H = \{h_1, \ldots, h_p\}$. Then we have $L \cup H \subseteq dom(\mu_{\mathrm{loc}_f})$. For the test case in [6, Fig.2], i.e. `DistCoord` in this work, H = {`x1,x2,y1,y2`} and L = {`k`}. We suppose that $\mu(e)$ is the evaluation of expression e under μ, and $(\mu, c) \Downarrow \mu'$ means that execution of c under μ derives μ'. We define the set-related indistinguishability $\mu =_L \mu'$ means $\forall \ell \in L.\mu(\ell) = \mu'(\ell)$, and $\mu =_{\{O,q\}} \mu'$ means $(q = q') \wedge (\forall 0 \leq i < q.O[i] = O'[i])$. The declassification-based privacy property is specified as follow.

Definition 2 (Privacy Preservation). *Data aggregation function f preserves privacy, iff* $\forall \mu_1, \mu_2 : (\mu_1 =_L \mu_2 \wedge \forall 0 \leq i < n.\mu_1(e_i) = \mu_2(e_i) \wedge (\mu_1, f_{body}) \Downarrow \mu_1' \wedge (\mu_2, f_{body}) \Downarrow \mu_2') \Rightarrow (\mu_1' =_{\{O,q\}} \mu_2')$*, where f_{body} is the command w.r.t. the body of f, and $e_i (0 \leq i < n)$ is (1) the expression which contains some $h_k \in H$ and is used to obtain the value of some explicit output variable, or (2) the expression explicitly released with declassification primitives.*

From the definition, we know the value of expression $e_i (0 \leq i < n)$ are what the LBS claimed to get, and we treat these expressions to be declassifiable. For `DistCoord`, the declassifiable expressions are {`(x2-x1)`^{2}, `(y2-y1)`^{2}, `(x1+x2)/2`, `(y1+y2)/2`}. The declassified values of these expressions are actually decided by the private location information and the LBS data which are passed to the aggregation function by parameters. The treatment of public input, i.e. variables in L, is different from that for delimited release [7]. We do not judge the indistinguishability on L at final states because even the LBS data passed by L can only be sent back to LBS through explicit output. The privacy property indicates that if the variation of confidential information (typically the private location information carried by parameters in H) cannot cause variation of initial value of e_i, then the variation of confidential information will not be detected through the public output. On the other hand, from the violation of property, i.e. variation of public outputs, we can learn that the leakage of confidential information is not caused by the declassifiable e_i.

2.3 Declassification Enforcement

We propose a reachability analysis to enforce the declassification-based privacy property. We develop our enforcement based on the model transformation algorithm in [11, Sec.IV]. With the transformation, the declassification-based privacy property can be reduced to a safety property that is enforceable using reachability analysis. The model is transformed with a variant of *compact self-composition* [10] facilitated with a set of auxiliary pushdown rules to perform the interleaving assignments and to construct the illegal-flow states. Also, the initial equivalences on declassifiable expressions are considered in these auxiliary pushdown rules.

Table 2. Compact Self-Composition

r	$\mathcal{SC}(r)$
$\langle\gamma_j\rangle \hookrightarrow \langle\gamma_k\rangle\ \mathcal{R}$	$\{\langle\gamma_j\rangle \hookrightarrow \langle\gamma_k\rangle\ \mathcal{R} \wedge rt(\xi(\mu \setminus \{O,q\})),$ $\langle\xi(\gamma_j)\rangle \hookrightarrow \langle\xi(\gamma_k)\rangle\ \mathcal{R}^{\xi(x)}_{x\in L\cup H} \wedge rt(\mu \setminus \{O,q\})\}$
$\langle\gamma_j\rangle \hookrightarrow_o \langle\gamma_k\rangle\ \mathcal{R}$	$\{\langle\gamma_j\rangle \hookrightarrow \langle\gamma_k\rangle\ \mathcal{R} \wedge rt(\xi(\mu \setminus \{O,q\})),$ $\langle\xi(\gamma_j)\rangle \hookrightarrow \langle\xi(\gamma_k)\rangle\ (O[q] = x) \wedge (q' = q+1) \wedge$ $rt(\mu \setminus \{q\}, \xi(\mu \setminus \{O,q\})),$ $\langle\xi(\gamma_j)\rangle \hookrightarrow \langle error\rangle\ (O[q] \neq x) \wedge rt(\mu, \xi(\mu \setminus \{O,q\}))\}$
$\langle\gamma_j\rangle \hookrightarrow \langle g_{\text{entry}}\gamma_k\rangle\ \mathcal{R}$	$\{\langle\gamma_j\rangle \hookrightarrow \langle g_{\text{entry}}\gamma_k\rangle\ \mathcal{R} \wedge$ $rt(\xi(\mu_{\text{glb}} \setminus \{O,q\})) \wedge rt_2(\xi(L\cup H)),$ $\langle\xi(\gamma_j)\rangle \hookrightarrow \langle\xi(g_{\text{entry}})\xi(\gamma_k)\rangle\ \mathcal{R}^{\xi(x)}_{x\in L\cup H} \wedge$ $rt(\xi(\mu_{\text{glb}} \setminus \{O,q\})) \wedge rt_2(L\cup H)\}$
$\langle\gamma_j\rangle \hookrightarrow \langle\epsilon\rangle\ \mathcal{R},$ $(\gamma_j \neq f_{\text{exit}})$	$\{\langle\gamma_j\rangle \hookrightarrow \langle\epsilon\rangle\ \mathcal{R} \wedge rt(\xi(\mu_{\text{glb}} \setminus \{O,q,iret\})),$ $\langle\xi(\gamma_j)\rangle \hookrightarrow \langle\epsilon\rangle\ \mathcal{R} \wedge rt(\xi(\mu_{\text{glb}} \setminus \{O,q,iret\}))\}$
$\langle f_{\text{exit}}\rangle \hookrightarrow \langle\epsilon\rangle\ \mathcal{R},$	$\{\langle\xi(f_{\text{exit}})\rangle \hookrightarrow \langle\xi(f_{\text{exit}})\rangle\ \mathcal{R}^{\xi(x)}_{x\in\mu\setminus\{O,q\}} \wedge rt(\mu \setminus \{O,q\})\}$

Table 3. Refinement of Auxiliary Pushdown Rules

PDS Rule	Refined PDS Rules
$\langle start\rangle \hookrightarrow \langle f_{\text{entry}}\rangle\ \mathcal{R}_1$	$\{\langle start\rangle \hookrightarrow \langle\gamma_{\text{mid}_1}\rangle\ \mathcal{R}_1,$ $\langle\gamma_{\text{mid}_1}\rangle \hookrightarrow \langle f_{\text{entry}}\rangle\ (\bigwedge_{0\le i<n} \mathcal{D}'[\rho(e_i)] = e_i) \wedge$ $rt(\mu, \xi(\mu \setminus \{O,q\}))\}$
$\langle f_{\text{exit}}\rangle \hookrightarrow \langle\xi(f_{\text{entry}})\rangle\ \mathcal{R}_2$	$\{\langle f_{\text{exit}}\rangle \hookrightarrow \langle\gamma_{\text{mid}_2}\rangle\ \mathcal{R}_2,$ $\langle\gamma_{\text{mid}_2}\rangle \hookrightarrow \langle\xi(f_{\text{entry}})\rangle\ (\bigwedge_{0\le i<n} \mathcal{D}[\rho(e_i)] = \xi(e_i)) \wedge$ $rt(\mu, \xi(\mu \setminus \{O,q\})),$ $\langle\gamma_{\text{mid}_2}\rangle \hookrightarrow \langle idle\rangle\ (\bigvee_{0\le i<n} \mathcal{D}[\rho(e_i)] \neq \xi(e_i))$

The compact self-composition $\mathcal{SC}$ used here is given in Table 2. ξ is the rename function on stack symbols to generate new flow graph nodes and on variables to generate companion variables for the model of second run. $\mathcal{R}^{\xi(x)}_{x\in V}$ is a relation derived by substituting each variable in V with the renamed companion variable. We annotate the abstraction of explicit output with $\hookrightarrow_o$, and then construct the illegal-flow state *error* within the compact self-composition by comparing the output x of the second run with the corresponding output stored in the first run to judge if they are not equal. $\mathcal{SC}$ is applied on each pushdown rule r in $\Phi(f_{\text{body}}, f_{\text{entry}}, f_{\text{exit}})$ to derive a set of pushdown rules $\bigcup_{r\in\Phi(f_{\text{body}},f_{\text{entry}},f_{\text{exit}})} \mathcal{SC}(r)$. The result of compact self-composition still needs two auxiliary pushdown rules:

1. $\langle start\rangle \hookrightarrow \langle f_{\text{entry}}\rangle\ \mathcal{R}_1$: This pushdown rule is used to perform initial interleaving assignments and to set the initial index q to 0;
2. $\langle f_{\text{exit}}\rangle \hookrightarrow \langle\xi(f_{\text{entry}})\rangle\ \mathcal{R}_2$: This pushdown rule is used to reset q for the reuse of output channel in the second run.

The initial interleaving assignments can be specified with

$$\mathcal{R}_1 \equiv (\forall \ell \in L.\ell' = \xi(\ell)) \wedge (q' = 0) \wedge rt(\mu \setminus \{L, q\}, \xi(\mu \setminus \{O, q\}))$$

and the index q can be reset with

$$\mathcal{R}_2 \equiv (q' = 0) \wedge rt(\mu \setminus \{q\}, \xi(\mu \setminus \{O, q\}))$$

Both pushdown rules are also involved in modeling the initial equivalence of declassifiable expressions. More specifically, the modeling of the initial $\mu_1(e_i) = \mu_2(e_i)$ leverages the store-match pattern [11] on auxiliary list $\mathcal{D}$. We define a function $\rho : \{e_i \mid 0 \leq i < n\} \mapsto \{0..n-1\}$ to map each declassifiable expression to the index of $\mathcal{D}$. Then we refine the above two auxiliary pushdown rules to add the store-match operations. The results are given in Table 3. For the case `DistCoord`, these refined pushdown rules are used to model

```
int f(...){
    k=$k; q=0;
    D[0]=(x2-x1)^2; D[1]=(y2-y1)^2; D[2]=(x1+x2)/2; D[3]=(y1+y2)/2;
    //model of the first run
    q=0;
    if(D[0]!=($x2-$x1)^2 || D[1]!=($y2-$y1)^2 ||
       D[2]!=($x1+$x2)/2 || D[3]!=($y1+$y2)/2) goto idle;
    //model of the second run
}
```

Reaching the state *idle* implies violation of some $\mu_1(e_i) = \mu_2(e_i)$, and whenever *idle* is reached, the illegal-flow state *error* becomes unreachable. Therefore the reachability of *idle* rules out the trace irrelevant to the judgement of post-condition of privacy preservation.

3 Evaluations

We modified our previous implementation [11] to support the privacy property verification. The aggregation functions are modeled with Remopla. The test cases are selected from related work to evaluate the preciseness of our enforcement compared with the global data-flow analysis [6], see Table 4. The length of DB we used for `PasswordChecker` is 100. `Density` and `CoordAvg` are manually implemented aggregation function. `Density` counts the number of nodes in a specific rectangle area, and `CoordAvg` calls `Avg` to derive the average coordinates of a set of nodes. The interfaces are like

```
int Density(int X[N],int Y[N],int up,int down,int left,int right);
int CoordAvg(int X[N],int Y[N]);
```

In Table 4, Non-Inf means the satisfaction of non-inference property, and GDF is the result of global data-flow analysis. RA is the result of our reachability analysis. *len* and *bit* are two factors of model that may have impact on the efficiency of our verification. *len* denotes the length of output channel, and *bit* is the number of bits for integer variables and elements in channel. T_{bit}^{len} is the time for reachability analysis of model with parameter *len* and *bit*. len_{min} is

Table 4. Precision (2.83GHZ×4 Intel CPU, 4GB RAM, Linux 2.6.38-8-generic)

Case	From	Non-Inf[6]	GDF	RA	len_{min}	bit_{min}	$T_{bit_{min}}^{len_{min}}$(s)	T_2^5(s)
DistCoord	Fig.2,[6]	✓	✓	✓	3	1	≤0.01	0.01
Avg(n=10)	Sec.3.3,[7]	✓	✓	✓	1	1	0.01	61.95
AvgAttack(n=10)	Sec.3.3,[7]	×	×	×	1	1	0.02	62.58
Wallet	Sec.3.3,[7]	✓	×	✓	1	1	≤0.01	≤0.01
WalletAttack	Sec.3.3,[7]	×	×	×	1	2	≤0.01	0.01
ParityAttack	Sec.3.4,[7]	×	×	×	1	2	≤0.01	≤0.01
Parity	Sec.3.4,[7]	✓	×	✓	1	1	≤0.01	≤0.01
ParitySec	Sec.4,[7]	✓	×	✓	1	1	≤0.01	≤0.01
PasswordChecker	Sec.5,[7]	✓	✓	✓	1	1	0.14	4.26
PasswordUpdate	Sec.5,[7]	✓	✓	✓	1	1	≤0.01	0.02
Density(n=100)	–	✓	✓	✓	1	1	0.06	37.66
CoordAvg(n=100)	–	✓	✓	✓	2	1	0.05	16.08

the minimum length of output channel that is sufficient to identify the possible inequality of output sequences, i.e. the violation of post-condition $\mu_1' =_{\{O,q\}} \mu_2'$ of privacy property. bit_{min} is the minimum *bit* that is required to avoid the false-positive. With a large enough *bit*, the false-positive is always avoidable therefore it will not influence the final preciseness of our approach. Meanwhile, our approach is more precise than the global data-flow analysis to avoid the false-negatives appeared in `Wallet`, `Parity`, and `ParitySec`. The false-negative is mainly cause by implicit information flow that can be avoid by our value-dependent approach. Although the global data-flow analysis is flow-sensitive and correctly judges the program `l=h1+h2;l=0` to be secure, the following program

```
if(h) {l=1} else {l=h+1}; output(l);
```

is mistakenly judged insecure by the global data-flow analysis. On the contrary, our value-dependent analysis can ensure it preserves privacy of the initial `h`.

From the comparison of $T_{bit_{min}}^{len_{min}}$ and T_2^5 we know the increase on value of either *len* or *bit* may cause the increase on the verification time. We use further experiment to evaluate the impact of *len* and *bit* on the efficiency of our verification. The impact is illustrated with the experimental results given in Fig. 2. The left side figures show the variation of verification time w.r.t. different length of output channel. In these experiments $bit = 2$ and the length of channel is set from 5 to 50 with common difference 5. We can see that the increase on length of output channel will not necessarily increase the verification time. The right side figure shows the variation of verification time when *bit* is increased and *len* is set to 5. We can see the verification time increases exponentially as the value of *bit* increases. Larger size of array leads to a faster increase on verification time. For example, `Avg`, `AvgAttack`, `Density`, `CoordAvg` and `PasswordChecker` cause a time limitation exceeded when $bit = 3$, while for `Parity`, `ParityAttack`, `ParitySec` and `WalletAttack`, we meet the TLE at $bit = 9$.

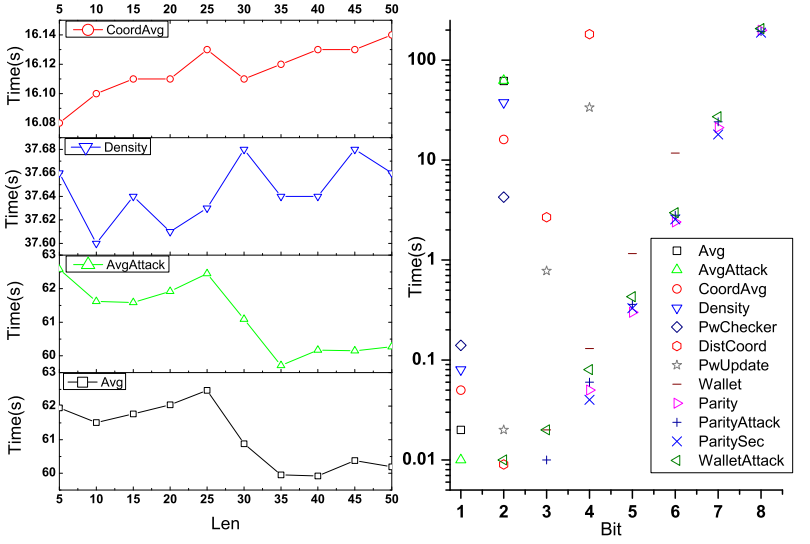

Fig. 2. Impact of *len* and *bit* on Efficiency of Verifications

References

1. Barthe, G., D'Argenio, P.R., Rezk, T.: Secure information flow by self-composition. In: CSFW, pp. 100–114. IEEE Computer Society (2004)
2. Beresford, A., Stajano, F.: Location privacy in pervasive computing. IEEE Pervasive Computing 2(1), 46–55 (2003)
3. Chow, C.Y., Mokbel, M.F.: Privacy in location-based services: a system architecture perspective. SIGSPATIAL Special 1(2), 23–27 (2009)
4. Hong, J.I., Landay, J.A.: An architecture for privacy-sensitive ubiquitous computing. In: MobiSys 2004, pp. 177–189. ACM, New York (2004)
5. Jiang, T., Wang, H.J., Hu, Y.C.: Preserving location privacy in wireless lans. In: MobiSys 2007, pp. 246–257. ACM, New York (2007)
6. Ravi, N., Gruteser, M., Iftode, L.: Non-inference: An information flow control model for location-based services. In: 3rd International Conference on Mobile and Ubiquitous Systems Workshops, pp. 1–10 (2006)
7. Sabelfeld, A., Myers, A.: A Model for Delimited Information Release. In: Futatsugi, K., Mizoguchi, F., Yonezaki, N. (eds.) ISSS 2003. LNCS, vol. 3233, pp. 174–191. Springer, Heidelberg (2004)
8. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Tech. Rep. SRI-CSL-98-04, SRI International (1998)
9. Schwoon, S.: Model Checking Pushdown Systems. Ph.D. thesis, Technical University of Munich, Munich, Germany (2002)
10. Sun, C., Tang, L., Chen, Z.: Secure information flow by model checking pushdown system. In: UIC-ATC 2009, pp. 586–591. IEEE Computer Society (2009)
11. Sun, C., Tang, L., Chen, Z.: A new enforcement on declassification with reachability analysis. In: INFOCOM 2011 Workshops, pp. 1024–1029. IEEE (2011)
12. Yiu, M.L., Jensen, C.S., Møller, J., Lu, H.: Design and analysis of a ranking approach to private location-based services. ACM Trans. Database Syst. 36, 1–42 (2011)