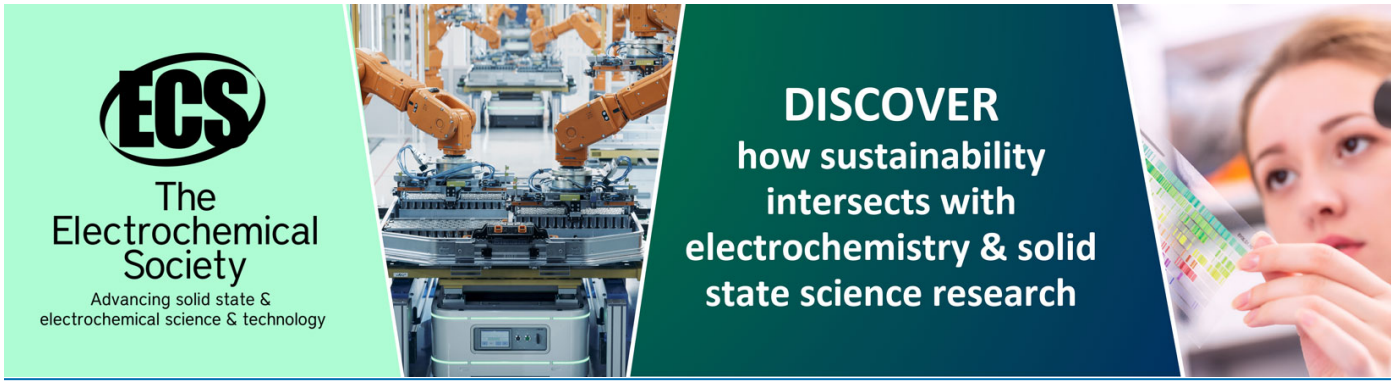
Forgery Attack on LightMAC Hash Function Scheme using SIMECK 32/64 Lightweight Block Cipher



IOP Conf. Series: Materials Science and Engineering **453** (2018) 012014 doi:10.1088/1757-899X/453/1/012014

Forgery Attack on LightMAC Hash Function Scheme using SIMECK 32/64 Lightweight Block Cipher

T A Darumaya[1] and B H Susanti[2]

[1,2]Sekolah Tinggi Sandi Negara, Ciseeng, Bogor 16120, Indonesia
[1]tifa.azzahra@student.stsn-nci.ac.id and [2]bety.hayat@bssn.go.id[2]

Abstract. LightMAC is one of MAC algorithm based on a block cipher that use two independent keys. As a MAC algorithm, it should be able to fulfill computation resistance. MAC algorithm is vulnerable to forgery attack if that property does not hold. There are three type of forgery attacks i.e., selective, rxistential, and universal forgery attack. In this paper, we do forgery attack on LightMAC scheme and use SIMECK 32/64 lightweight block cipher as the basic construction. We use Wang et al. method's for selective forgery and Fanbou & Fengmei method's for universal forgery, whereas the method of existential forgery was determined based on observation of the structure. The attack result shows that we can get the forged message for every sample. The value of the forged messages is the same for all the type of forgery attacks. So, LightMAC is vulnerable to forgery attack.

1. Introduction

Hash function is a cryptographic technique that maps bit-strings of arbitrary finite length to strings of fixed length [1]. It can be used to ensure the integrity of data [1]. There are two types of hash function, i.e. Messages Authentication Codes (MAC) and Messages Detection Codes (MDC). The difference between these types is the input parameter. MAC uses a secret key and a message as the input, while MDC only uses a message. In this research, we will focus on the MAC algorithm. There are three properties that should be fulfilled in MAC such as ease of computation, compression, and computation resistance.

Nowadays, lightweight cryptography is the one of the research that has been developed in cryptography. Poschmann classified it into lightweight block cipher, lightweight hash function, and lightweight public key [2]. SIMECK is an example of lightweight block cipher based on Feistel structure designed by Yang *et al.* [3]. In 2016, Luykx *et al.* designed a lightweight hash function based on block cipher called LightMAC. LightMAC can be used as a MAC algorithm and pseudorandom function [4].

Menezes *et al.* claimed that if computation resistance does not hold, MAC algorithm is vulnerable to MAC forgery [1]. This statement has been proven by some researchers who did forgery attack on some MAC algorithm. In 2011, Wang *et al.* did a security evaluation on PC-MAC that use Advanced Encryption Standard as the basic construction. Among all of the attacks they did, there are selective, existential, and universal forgery attack too [5]. Meanwhile, universal forgery attack to some block cipher-based MAC and Authenticated Encryption was performed by Fanbao & Fengmei with birthday paradox [6]. They claimed that LightMAC is vulnerable to universal forgery attack. But they did not show the implementation and results according to their claim.

Based on some facts and the previous research, we will conduct a research about the application of selective, existential, and universal forgery attack on LightMAC scheme of SIMECK 32/64 algorithm. SIMECK 32/64 is the lightweight block cipher that will be used as a building block cipher

due to limited available computation resistance. The purpose of this research is to know the resistance of LightMAC scheme against three types of forgery attack based on SIMECK 32/64. The results can be used for consideration when we will implement LightMAC scheme on a platform.

2. Related Theories

2.1. Messages Authentication Codes

MAC is a keyed hash function h_k that uses a secret key k as an input. MAC are used between two parties that share a secret key to authenticate information exchanged between them [7]. MAC provides a message's integrity and authenticity by creating a tag value [8], with the following properties [1]:

- Ease of computation
  Given a value k and input x for a known function h_k, $h_k(x)$ is easy to compute.
- Compression
  Hash function h_k maps an input x of arbitrary finite bit length to an output $h_k(x)$ of fixed bit length n.
- Computation resistance
  Given zero or more text-MAC pairs $(x_i, h_k(x_i))$ it is computationally infeasible to compute any text-MAC pair $(x, h_k(x))$ for any new input that fulfills $h_k(x) = h_k(x_i)$

2.2. LightMAC hash function scheme

LightMAC is a lightweight hash function based that can be used as either a pseudorandom or a MAC. The parameters of LightMAC are the integers s and t, i_s, and the block cipher E with fixes key k and n bit length input. The tag value denoted by t and i_s is a representation of a counter with s bit length. For details, we refer to [4].

2.3. SIMECK lightweight block cipher

SIMECK is a lightweight block cipher that combines the component from both SIMON and SPECK. It can be denoted as SIMECK $2n/mn$ where n is the word size and mn is the key size. More specific, it includes SIMECK $32/64$, SIMECK $48/96$, and SIMECK $64/128$. See [3] for more details discussion.

2.4. Forgery attack

MAC algorithm is considered secure if attacker is not able to create a tag of a message without knowing the key. Such a made-up message/tag pair is called a forgery. On the other hand, attempting to recover a key is just a specific case of a more general class of attacks is called forgery attack [8]. There are three type of forgery attacks such as selective forgery, existential forgery, and universal forgery [9]. Each forgery attack depends on the used assumption. Selective forgery happens if an adversary is able to produce a new input-tag value pair of his input choice. Whereas, existential forgery has no control over the tag value of an input to produce a new input-tag value pair. Universal forgery has the condition if an adversary is able to find a tag value for every given message [6].

3. Research Metodology

In this research, the parameters that we use for LightMAC scheme are $s = 16$ bit, $t = 32$ bit, $i_s = 16$ bit. SIMECK 32/64 as the block cipher with fixes k_i for $i \in 1, 2$ and $n = 32$ bit. The value of k_1 is $246660af066f3f15_H$ whereas k_2 is $2f213252b9cc686e_H$. The size of the message M is 60 bit, so we divided it into 4 block messages and each block use 16 bit plaintext. For the last block, LightMAC will process 12 bit of plaintext and concate it with padding of string 10000_H. This parameter will be used in all types of the forgery attack.

This experiment performed by implementing SIMECK 32/64 lightweight block cipher into LightMAC scheme using C programming language and Dev C++ compiler. Then, we do forgery attacks to it. We use five random inputs and five non-random inputs as the sample which has 60 bit length. The

modification of the block message was done in two ways. For selective and universal forgery attack, we will generate it randomly using C programming language as much as 2^{16}. While, for existential forgery attack, we do the modification by increment from 0000_H until $ffff_H$. The steps of the forgery attacks explained in 3.1 until 3.3.

3.1. Selective forgery attack

The selective forgery attack's method refers to Wang *et al.*'s [5]. Assumed that the adversary can choose the messages $M = m_1 \| m_2 \| m_3 \| m_4$ and generate the hash value $MAC_k(M) = C$. M is the message to be forged. This attack can be implemented as follows:

- Generate $2^{n/2}$ random message in the second block m_2^i where $0 \le i \le 2^{n/2}$.
- Calculate the value of $a_i = E_k(1_s \| m_1) \oplus E_k(2_s \| m_2^i)$.
- Generate $2^{n/2}$ random message in the first block m_1^j where $0 \le j \le 2^{n/2}$.
- Calculate the value of $b_j = E_k(1_s \| m_1^j) \oplus E_k(2_s \| m_2)$
- Compare the value of and b_j, if $a_i = b_j$ query $m_1^j \| m_2^i$ to $M^{'}$.
- Calculate $MAC_k(M^{'}) = C^{'}$ and obtain $C^{'}$ would be valid if compared with C.

3.2. Existential forgery attack

Given a message $M = m_1 \| m_2 \| m_3 \| m_4$ and the corresponding hash value $MAC_k(M) = C$, the adversary can do the attack as follows:

- Calculate the value $c = E_k(1_s \| m_1) \oplus E_k(2_s \| m_2)$.
- Generate $2^{n/2}$ message in the first block m_1^i of LightMAC scheme where $0 \le i \le 2^{n/2}$.
- Calculate the encryption result of the first block message $a_i = E_k(1_s \| m_1^i)$.
- Generate $2^{n/2}$ message in the second block m_2^j of LightMAC scheme where $0 \le j \le 2^{n/2}$
- Calculate the encryption result of the second block message $b_j = E_k(2_s \| m_2^j)$.
- Calculate the value of $z_t = a_i \oplus c$ where $0 \le t \le 2^{n/2}$.
- Determine z_t that statisfy $z_t = b_j$.
- Query $m_1^i \| m_2^j$ to $M = m_1^i \| m_2^j \| m_3 \| m_4$.
- Calculate $C^{'} = MAC_k(E_k(1_s \| m_1^i) \oplus E_k(2_s \| m_2^j) \oplus E_k(3_s \| m_3) \oplus E_k(4_s \| 10^*))$. The hash value $C^{'}$ would be valid if compared with C.

3.3. Universal forgery attack

The universal forgery attack method refers to Fanbou & Fengmei's [6]. For any given messages $M = m_1 \| m_2 \| m_3 \| m_4$ and the corresponding hash value $MAC_k(M) = \tau$, the adversary can forge the message as follow:

- Randomly generate $2^{n/2}$ on the second block messages m_2^i where $0 \le i \le 2^{n/2}$.
- Calculate the hash value $\tau_i = MAC_k(E_k(1_s \| m_1) \oplus E_k(2_s \| m_2^i) \oplus E_k(3_s \| m_3) \oplus E_k(4_s \| 10^*))$ in group G_1.

- Randomly generate $2^{n/2}$ on the first block messages m_1^j where $0 \le j \le 2^{n/2}$.
- Calculate the hash value $\tau_j = MAC_k(E_k(1_s \| m_1^j) \oplus E_k(2_s \| m_2) \oplus E_k(3_s \| m_3) \oplus E_k(4_s \| 10^*))$ in group G_2.
- Compare the hash value τ_i and τ_j, if it statisfy $\tau_i = \tau_j$ query the message $m_1^j \| m_2^i$.
- Calculate the hash value $\tau' = MAC_k(E_k(1_s \| m_1^j) \oplus E_k(2_s \| m_2^i) \oplus E_k(3_s \| m_3) \oplus E_k(4_s \| 10^*))$ and it would be valid if compared with τ.

4. Forgery attack on LightMAC scheme using SIMECK 32/64

After the application of selective, existential, and universal forgery attacks against LightMAC scheme using SIMECK 32/64, we can obtain the collisions for each attack. Table 1 and Table 2 show the collision results or we called it as a forged message for random inputs and non-random inputs in hexadecimal representation. Sample Message column showed all of the random sample that we use to be forged. Forge Message column showed the forged message that we obtained. Selective Forgery column showed the values of a_i and b_j that have the same value. Existential Forgery column showed the values of and b_j that statisfy Universal Forgery column showed the MAC value of group G_1 and G_2. MAC Value column showed the valid MAC value.

Table 1. Forgery Attack on Random Inputs

Nu.	Sample Message	Forge Message	Selective Forgery	Existential Forgery	Universal Forgery	*MAC* Value
1.	$a1e24baa18f8687$	$188e0f8218f8687$ $122a08af18f8687$	$dfaaf7b0$ $bd9ea1ac$	$6d7baa82$ $0f4ffc9e$	$caa0ec31$ $af05e685$	$3118a82a$
2.	$285954b7f878c4c$	$b2b5b598f878c4c$	$2b0ed874$	$3168099a$	$ab4ce512$	$a9c99f8a$
3.	$f5a14bb1a7de9f5$	$0128ad49a7de9f5$ $c8f2a2b3a7de9f5$	$4a342801$ $0fb3e334$	$324da738$ $77ca6c0d$	$a1c337e8$ $12b9a225$	$aa8364fa$
4.	$c1fb41d8be3dede$	$44deb2bdbe3dede$	$27aa1b60$	$fb596736$	$980e9c4f$	$10ca185a$
5.	$cfa7c8845634b40$	$33c163955634b40$ $2ff767815634b40$	$0962567d$ $ac75f44c$	$b2487270$ $175fd041$	$8b768b82$ $6d9634a5$	$65ab94f3$
		Total of Collision				**9**

From Table 1, we can see that the forged message can be obtained for every samples. Based on the method, we just do the modification on the first and second block messages. So the third and fourth messages have the same value. The forged messages for all type of forgery attack has the same value. For example, the first sample is $a1e24baa18f8687_H$ has two forged messages that are $188e0f8218f8687_H$ and $122a08af18f8687_H$ for all types of forgery attack. The difference between it depends on the value being compared and will be explained in this section.

Selective forgery attack compare the value of a and b. Based on the steps in section 3.1, we do 2^{16} modification of the first block message m_1^j and second block message m_2^i.So, we have 2^{16} values of a and 2^{16} values of b. For example, we will explain this attack with the first sample message is $a1e24baa18f8687_H$ with a MAC value is $3118a82a_H$. Then we do modification in the second block and find the value of a is (shown at Figure 1):

$$a = E_k(1_s \| a1e2_H) \oplus E_k(2_s \| 0f82_H) \quad (1)$$
$$= b2d15d32_H \oplus 6d7baa82_H$$
$$= dfaaf7b0_H$$

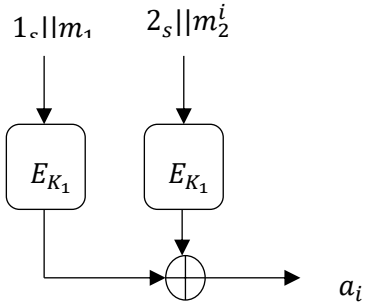


Figure 1. Selective forgery attack (the value of a)

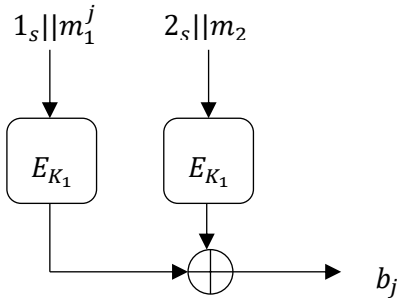


Figure 2. Selective forgery attack (the value of b)

After that, we do modification on the first block message and calculate the value of b (shown at Figure 2):

$$b = E_k(1_s \| 188e_H) \oplus E_k(2_s \| 4baa_H) \quad (2)$$
$$= 67a05cdb_H \oplus b80aab6b_H$$
$$= dfaaf7b0_H$$

As we can see, the value of a and b is the same, so we calculate the tag value with the new input message of $188e0f8218f8687_H$. The results of MAC value produced in this attack is $3118a82a_H$. It is valid with the original message, so the input of $188e0f8218f8687_H$ was called forged message.

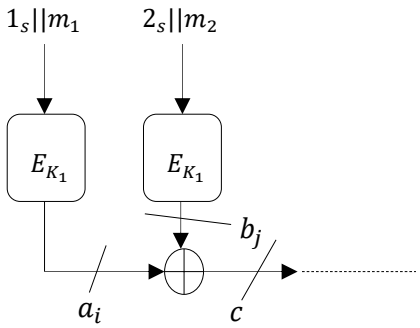


Figure 3. Existential forgery attack

Existential forgery would compare the value of z and b and refers to the steps in section 3.2. We will use the first sample message for the explanation. See the Figure 3 for the scenario of the attack. We calculate the value of c as:

$$c = E_k(1_s \| a1e2_H) \oplus E_k(2_s \| 4baa_H) \quad (3)$$
$$= b2d15d32_H \oplus b80aab6b_H$$
$$= 0adbf659_H$$

After that, we modified the first block message and set the value of a. We find one of the a value for this example:

$$a = E_k(1_s \| 188e_H) = 67a05cdb_H$$

Then, we modify the second block message and calculate the value of b :

$$b = E_k(2_s \| 0f82_H) = 6d7baa82_H$$

Last , we calculate the value of z :

$$\begin{aligned} z &= E_k(1_s \| 188e_H) \oplus c \\ &= 67a05cdb_H \oplus 0adbf659_H \\ &= 6d7baa82_H \end{aligned}$$

So, we get the forged message $188e0f8218f8687_H$ for input $a1e24baa18f8687_H$. The MAC value for this sample $3118a82a_H$. Furthermore, we will explain application of universal forgery attack based on section 3.3. It compare the MAC value of group G_1 and group G_2 Figure 4 is the illustration of universal forgery attack's scenario.

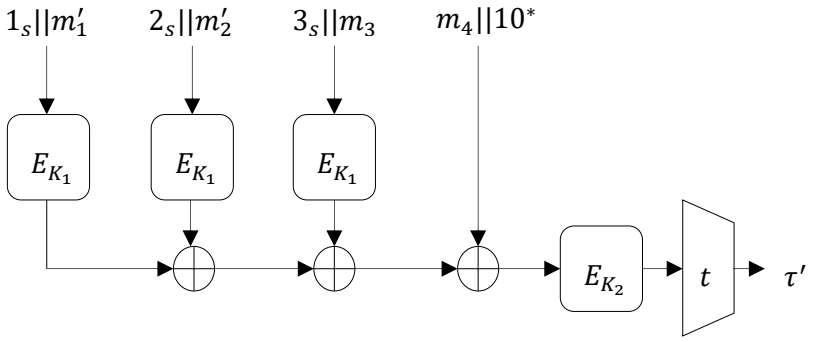


Figure 4. Universal forgery attack

We use the first sample message from Table 1 is $a1e24baa18f8687_H$ for the explanation. First we generate 2^{16} modification on the second block message $m_2^{'}$ and calculate the MAC value. This modification was done in group G_1. For example:

$$\begin{aligned} \tau_i &= MAC_k(E_k(1_s \| m_1) \oplus E_k(2_s \| m_2^{'}) \oplus E_k(3_s \| m_3) \oplus (m_4 \| 10^*)) \\ &= MAC_k(E_k(1_s \| a1e2_H) \oplus E_k(2_s \| 0f82_H) \oplus E_k(3_s \| 18f8_H) \oplus (68780000_H)) \\ &= MAC_k(67a05cdb_H \oplus 6d7baa82_H \oplus d9065856_H \oplus 68780000_H) \\ &= caa0ec31_H \end{aligned}$$

Then, we generate 2^{16} modification on the first block message m_1' and calculate the MAC value of group G_2. For example, we use $188e_H$ as m_1' and calculate it:

$$\tau_j = MAC_k(E_k(1_s \| m_1^{'}) \oplus E_k(2_s \| m_2) \oplus E_k(3_s \| m_3) \oplus (m_4 \| 10^*))$$

$$= MAC_k(E_k(1_s \| 188e_H) \oplus E_k(2_s \| 4baa_H) \oplus E_k(3_s \| 18f8_H) \oplus (68780000_H))$$
$$= MAC_k(67a05cdb_H \oplus b80aab6b_H \oplus d9065856_H \oplus 68780000_H)$$
$$= caa0ec31_H$$

We get τ_i and τ_j that statisfy $\tau_i = \tau_j$, so we query the message $m_1' || m_2'$ and calculate the MAC value as follows:

$$\tau' = MAC_k(E_k(1_s \| m_1') \oplus E_k(2_s \| m_2') \oplus E_k(3_s \| m_3) \oplus (m_4 \| 10^*))$$
$$= MAC_k(E_k(1_s \| 188e_H) \oplus E_k(2_s \| 0f82_H) \oplus E_k(3_s \| 18f8_H) \oplus (68780000_H))$$
$$= MAC_k(67a05cdb_H \oplus 6d7baa82_H \oplus d9065856_H \oplus 68780000_H)$$
$$= 3118a82a_H$$

The value of τ' would be valid with τ, then the forged message for M is $188e0f8218f8687_H$. We also use the similiar explanation for non-random inputs (See Table 2).

Table 2. Forgery Attack on Non-Random Inputs

Nu.	Sample Message	Forge Message	Selective Forgery	Existential Forgery	Universal Forgery	*MAC* Value
1.	111111111111111	64f9cfbb1111111 e92505fe1111111	4861cd61 c64bb976	30feabb2 bed4dfa5	55afddf5 71409c8f	7bfe34a6
2.	222222222222222	3d00a3702222222	f59ccda7	5aa9944a	6a38c6bd	5b149ecb
3.	333333333333333	-	c979 7667	9e34 d234	9895 9fb6	98959fb6
4.	444444444444444	-	552a a15c	7664 35dc	766435dc	766435dc
5.	555555555555555	93008eda5555555 05f78be45555555	488bd3da d3d9aa4d	5c41e4e1 c7139d76	33112c0c 1590f4d7	d190e646
		Total of Collision				**5**

From Table 1 and 2, we found a condition that the forged message of all the types of forgery attack was the same even when we use a different method for each forgery attack. After we analyze it, there are some reasons of LightMAC that cause this condition. First, identical tag value results when the same plaintext is encrypted under the same key k_1 and k_2 of LightMAC scheme. So, we have the same object for each sample in all of the forgery attacks that we will use. Next, the chaining mechanism causes a ciphertext depends on preceding block where the counter parameter in each block will affect the ciphertext's value. Based on the method of the forgery attack, we just modified the first and second block message so we have the same target to attack for each forgery types. Finally, total modification for each block is 2^{16} messages. It means that we have 32 bits plaintext where 16 bits most significant bit is a counter and 16 bit least significant bit is a plaintext, which is the population of the modification. As the result, we got the same forged message for every type of forgery attack.

5. Conclusion

In this study, we conduct three types of forgery attacks on LightMAC scheme which use SIMECK 32/64 as the basic construction. The results show that we can obtain the forged message for each sample by 2^{16} modification in each type of attacks. There are some reasons that influence the forge message's vaue, i.e. an identical tag value, the chaining mechanism, and the population of modification. When we implementing LightMAC scheme in a platform, this results should be taken for consideration. In future, further research needs to be done to the other MAC schemes with another block cipher algorithms as the basic contruction to examine their computation resistance property.

Acknowledgements
This study was supported by Sekolah Tinggi Sandi Negara, Bogor, West Java. We would like to thank the reviewers who give their valuable comments.

References
[1] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Applied Cryptography,* vol. Menezes, A, 1997.
[2] A. Poschmann, *Lightweight Cryptography: Cryptographic Engineering for a Pervasive World*, *Ph. D. Thesis*, no. February, pp. 1–197, 2009.
[3] G. Yang, B. Zhu, V. Suder, M. D. Aagaard, and G. Gong, *The Simeck Family of Lightweight Block Ciphers*, June, 2015.
[4] A. Luykx, B. Preneel, E. Tischhauser, and K. Yasuda, *A MAC Mode for Lightweight Block Ciphers*, 2017.
[5] L. Wang, K. Sakiyama, I. Nishikado, and K. Ohta, *Security Evaluation of PC-MAC-AES*, 2011.
[6] F. Liu and F. Liu, *Universal Forgery with Birthday Paradox : Application to Blockcipher-based Message Authentication Codes and Authenticated Encryptions*, pp. 1–24, 2017.
[7] W. Stallings, *Cryptography and Network Security Principles and Practices*. 2017.
[8] J. Aumasson, *Serious Cryptography*. No Starch Press, Inc., 2018.
[9] K. Jia, X. Wang, Z. Yuan, and G. Xu, *Distinguishing and second-preimage attacks on CBC-like MACs*, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5888 LNCS, no. 60525201, pp. 349–361, 2009.