Adapting the Weaknesses of the Random Oracle Model to the Generic Group Model

Alexander W. Dent*

Information Security Group,
Royal Holloway, University of London,
Egham Hill, Egham, Surrey, UK.
alex@fermat.ma.rhul.ac.uk
http://www.isg.rhul.ac.uk/~alex/

Abstract. The generic group model has recently been used to prove the security of certain asymmetric encryption and signature schemes. This paper presents results that show that there exist problems in that are provably hard in the generic group model but easy to solve whenever the random encoding function is replaced with a specific encoding function (or one drawn from a specific set of encoding functions). In particular we show that there exist cryptographic schemes that are provably hard in the generic group model but easy to break in practice.

1 Introduction

The complex nature of asymmetric encryption schemes makes it difficult to give concrete assurances of their security. In order to prove results about their security several models have been proposed. Each model makes some assumptions about the properties of certain parts of the scheme.

The most popular of these is the random oracle model, which was introduced by Bellare and Rogaway in 1993 [1]. It was designed to show the difficulty of breaking cryptographic algorithms by modelling certain parts of the cipher (usually the hash functions) as random functions. Doubt was cast on the validity of this model by Canetti, Goldreich and Halevi [3] who proved that there exists a theoretical signature scheme that is secure in the random oracle model but insecure when the random function is replaced by any polynomial time computable function or set of functions.

The generic group model was proposed by Shoup [8] to give exact bounds on the difficulty of the discrete logarithm problem and the Diffie-Hellman problem in the situation where the attacker has no information about the specific representation of the group being used. In other words the attacker is trying to solve a discrete logarithm (or Diffie-Hellman) problem in a group isomorphic to C_p but does not know whether this group is realised as, say, a multiplicative group or as an elliptic curve group. We cast some doubt on the model by proposing

* The work described in this paper has been supported by the Commission of the European Communities through the IST program under contract IST-1999-12324.

a problem that is provably difficult in the generic group model but for which there exists an attacker that can easily solve the problem for any representation of the group without using any properties of the special properties of that representation.

More recently the generic group model has been used by Brown [2], Schnorr and Jakobsson [7], and Smart [9] in the analysis of certain cryptographic protocols based on the Diffie-Hellman problem. Our result shows that, in the analysis of asymmetric primitives, the generic group model has the same weaknesses as the random oracle model. In particular we show how a secure signature scheme may be modified to give a scheme that is still secure in the generic group model but insecure whenever any specific representation of the group is chosen.

This work is similar in its intent to the work of Fischlin [4] but our result is an improvement. Fischlin shows that the security of the Schnorr signature scheme [6] in the generic group model might depend upon the choice of hash function used within the scheme. The paper shows that the scheme is weak in the generic group model with one particular hash function and postulates that the scheme is secure in the generic group model with a different hash function. We improve upon this result and show that if there exists any signature scheme that is secure in the generic group model then there exists a tweaked version of that scheme that is still provably secure in the generic group model but insecure in practice.

2 The Generic Group Model

Let p be a k-bit prime and let $\mathbb{Z}_p$ be the group of additive integers modulo p. Let $l_{\text{out}} : \mathbb{N} \to \mathbb{N}$ be a length function with $l_{\text{out}}(k) \geq k$ and $S = \{0,1\}^{l_{\text{out}}(k)}$. Note that it is possible to represent elements of $\mathbb{Z}_p$ as members of S. An encoding function is a function $\sigma : \mathbb{Z}_p \to S$ for which $\sigma(x) = \sigma(y)$ if and only if $x = y$.

The most common examples of encoding functions include representing an element $x \in \mathbb{Z}_p$ as:

- the bit representation of x in $\mathbb{Z}_p$,
- the bit representation of g^x in $\mathbb{Z}_m$, where g has order p in $\mathbb{Z}_m$,
- the bit representations of the co-ordinates of the elliptic curve point xP, where P is a point of order p on an elliptic curve E.

It is important to note that finding x given $\sigma(x)$ and $\sigma(1)$ is the same as solving the discrete logarithm problem on the group.

A generic algorithm is a probabilistic, polynomial-time Turing machine M that takes representations of group elements $\sigma(x_1), \dots, \sigma(x_m)$ as inputs. As M is executed it may compute group operations on group elements by way of an addition oracle $\mathcal{O} : S \times S \times \mathbb{Z}_2 \to S$ such that

$$\mathcal{O}(\sigma(x_i), \sigma(x_j), b) = \sigma(x_i + (-1)^b x_j) \ . \tag{1}$$

We assume that any call to this oracle involves one evaluation of σ.

We will denote a generic algorithm M with access to an encoding function σ and a suitable addition/subtraction oracle by M^σ (we implicitly assume the presence of an addition oracle whenever we have an oracle for the encoding function). We define the result of running such an algorithm as $x \leftarrow M^\sigma$. This differs from the original definition of [8] as our generic algorithm can calculate $\sigma(x)$ for any $x \in \mathbb{Z}_p$ without calculating any intermediate values. In particular M can calculate $\sigma(1)$.

This does not substantially change any of the results given in [8] because even when it is not possible to calculate $\sigma(x)$ directly, it is always possible to calculate $\sigma(x)$ using a polynomial number of queries to the addition oracle $\mathcal{O}$. The following is a result of Shoup [8].

Result 1. *Let $x \in \mathbb{Z}_p$ and let σ be a randomly chosen encoding function of $\mathbb{Z}_p$ into S. If M^σ is a generic algorithm for $\mathbb{Z}_p$ on S that makes at most m evaluations of σ then the probability that $x \leftarrow M^\sigma(\sigma(x))$ is $O(m^2/p)$. Note that in particular if M^σ makes a number of evaluations of σ that is polynomial in k then the probability that $x \leftarrow M^\sigma(\sigma(x))$ is negligible.*

3 Evasive Relations on Groups

The definition of an evasive relationship was introduced in [3] and we will continue to develop definitions and use proof techniques that that paper suggests. The notion of evasive relations capture one difference between random functions (a function chosen at random from all possible functions) and functions actually used in practice (that must be calculatable).

Definition 1 (Evasive Relation). *A relation $R \subseteq \{0,1\}^* \times \{0,1\}^{l_{\text{out}}(k)}$ is said to be* evasive *if for any probabilistic polynomial-time Turing machine M with access to an oracle $\mathcal{P}$ we have*

$$Pr[x \leftarrow M^{\mathcal{P}}(1^k),\ (x, \mathcal{P}(x)) \in R]$$

is negligible in k, where the probability is taken uniformly over all choices of oracle $\mathcal{P} : \{0,1\}^* \rightarrow \{0,1\}^{l_{\text{out}}(k)}$ and the coins of M.

We extend this definition so that it is applicable to the group setting.

Definition 2 (Evasive Group Relation). *A relation $R \subseteq G \times S$ is said to be an* evasive group relation *if for any probabilistic polynomial-time Turing machine M we have*

$$Pr[x \leftarrow M^\sigma(1^k),\ (x, \sigma(x)) \in R]$$

is negligible in k, where the probability is taken uniformly over all choices for an encoding function $\sigma : G \rightarrow S$ and the coins of M.

However, in the real world we will not be working with a random encoding function but with a known computable function that is, at worst, chosen from some collection. For example we could be working in a subgroup of the multiplicative group of integers modulo a value or a subgroup of an elliptic curve group with the points represented as either compressed, uncompressed or hybrid bit-strings. We designate the collection of these possible encoding functions an "encoding ensemble".

Definition 3 (Encoding ensemble). *We define an* encoding ensemble $\mathcal{F}$ *to be a collection of encoding functions* $f_s : \mathbb{Z}_p \to S$ *where* $s \in \{0,1\}^k$*. (We do not require that* $\mathcal{F}$ *contain exactly* 2^k *functions, just that this is an upper bound). We require that there exists a polynomial-time algorithm* Eval *such that* $\textsc{Eval}(s, x) = f_s(x)$ *and a polynomial-time algorithm* Add *such that*

$$\textsc{Add}(s, f_s(x_i), f_s(x_j), b) = f_s(x_i + (-1)^b x_j) \ . \tag{2}$$

Once again we reiterate the fact that complete knowledge of an encoding function f_s and the encoding a group element $f_s(x)$ does not imply that it is feasible to calculate x. This is the discrete logarithm problem and in general this is hard. However, in general, it is not necessary to be able to invert an encoding function in order to construct the Add function - all of the examples of encoding functions given in Section 2 have efficient Add functions, even when the discrete logarithm problem is thought to be hard for that representation.

We try to emulate the idea of evasive group relation when the randomly chosen encoding function is replaced with a function chosen at random from an encoding ensemble.

Definition 4 (Correlation intractability). *Let* $\mathcal{F}$ *be an encoding ensemble of* $\mathbb{Z}_p$ *into* S*.* $\mathcal{F}$ *is* correlation intractable *if for every probabilistic, polynomial-time Turing machine* M *and every evasive group relation* R *we have that*

$$Pr[s \leftarrow \{0,1\}^k,\ x \leftarrow M(s),\ (x, f_s(x)) \in R]$$

is negligible in k*, where the probability is taken over the uniformly random choice of* s *and the coins of* M*.*

A clear example of the difference between random encoding functions (an encoding function drawn at random from all possible encoding functions) and encoding ensembles (where the encoding function is drawn from a specific set) is that there exists no encoding ensemble which is correlation intractable.

Lemma 1. *There exist no correlation intractable encoding ensembles.*

Proof. Let $\mathcal{F}$ be an encoding ensemble of $\mathbb{Z}_p$ into S and define the relation R to be

$$R = \{(\bar{s}, f_s(\bar{s})) : s \in \{0,1\}^k\} \tag{3}$$

where $\bar{s} = s \pmod p$. This is an evasive relation because for every $x \in \mathbb{Z}_p$ there exists at most two y such that $(x, y) \in R$ and so, for any $x \in \mathbb{Z}_p$, we have that

$$Pr[(x, \sigma(x)) \in R] \leq \frac{1}{2^{l_{\text{out}}(k)-1}} \leq \frac{1}{2^{k-1}} \tag{4}$$

for a randomly chosen encoding function σ.

However if $M(s)$ is the machine that returns $\bar{s}$ then

$$Pr[s \leftarrow \{0,1\}^k,\ \bar{s} \leftarrow M(s),\ (\bar{s}, f_s(\bar{s})) \in R] = 1 \tag{5}$$

for any random choice of $s \in \{0,1\}^k$. So $\mathcal{F}$ is not a correlation intractable encoding ensemble. □

4 A Hard Problem with an Easy Solution

In this section we will examine a slightly modified version of the discrete logarithm problem. We still attempt to solve the discrete logarithm problem in the group $\mathbb{Z}_p$ as it is represented by the encoding function, however we now allow the attacking machine to have access to certain oracles. We attempt to show that whilst this problem is secure in the generic group model, it is insecure whenever any specific encoding function or encoding ensemble is used.

4.1 A Modified Problem

For any evasive group relation R we define an oracle D_R^σ such that

$$D_R^\sigma(y, \sigma(x)) = \begin{cases} x & \text{if } (y, \sigma(y)) \in R, \\ \perp & \text{otherwise.} \end{cases} \tag{6}$$

We still have that

Theorem 2. *If M^{σ, D_R^σ} is a generic algorithm that makes at most a number of queries to any oracle that is polynomial in n then*

$$Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x))]$$

is negligible, where the probability is taken over the uniform choice of encoding function σ and the coins of M.

Proof. Obviously the oracle D_R^σ does not affect M unless it is queried with a value y such that $(y, \sigma(y)) \in R$). Since R is a group evasive relation this probability is negligible, hence we may ignore the oracle D_R^σ. However in this case we may appeal to Result 1, which proves that the probability of M^σ returning x without the oracle D_R^σ is also negligible.

Formally we define E to be the event that the oracle D_R^σ is queried with (y, z) such that $(y, \sigma(y)) \in R$ and $\bar{E}$ be the complement of this event. So,

$$\begin{aligned} Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x))] &= Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x)) | E] Pr[E] \\ &\quad + Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x)) | \bar{E}] Pr[\bar{E}] \\ &\leq Pr[E] + Pr[x \leftarrow M^{\sigma, D_R^\sigma}(\sigma(x)) | \bar{E}] \end{aligned} \tag{7}$$

and both of these terms are negligible, the latter by Result 1. □

This proves that the oracle D_R^σ has no effect on the problem in the generic group model. Now consider the effects of this oracle when the random encoding function σ is replaced by an encoding ensemble. (Or rather the function σ chosen at random from all encoding functions is replaced with a function f_s chosen at random from the encoding ensemble $\mathcal{F}$.) If we use the group evasive relation R defined in (3) then the previously useless oracle D_R^σ now becomes

$$D_R^s(y, f_s(x)) = \begin{cases} x & \text{if } (y, f_s(y)) \in R, \\ \perp & \text{otherwise.} \end{cases} \tag{8}$$

Of course now there exists a machine $M^{D_R^s}(f_s(x), s)$ that will output x with probability 1 just by querying the oracle D_R^s with the query $(\bar{s}, f_s(x))$, where $\bar{s} \in \mathbb{Z}_p$ and $\bar{s} \equiv s \bmod p$.

4.2 A Universal Encoding Ensemble

So far we have shown that for every encoding ensemble there exists an oracle discrete logarithm problem that is provably difficult in the generic group model but easy when the random encoding function is replaced by a specific given encoding ensemble. We will now attempt to generalize this to an oracle discrete logarithm problem that is hard in the generic group model but easy when the random encoding function is replaced by *any* encoding ensemble. In order to do this we will need to enumerate all possible encoding ensembles.

Recall that for any function ensemble $\mathcal{F}$ there exists a polynomial-time function $\textsc{Eval}(s, x)$ that evaluates $f_s(x)$. We cannot enumerate all polynomial-time functions as there is no single polynomial-time bound that they all obey, so instead we enumerate all functions that run in time $t(k) = k^{\log k}$. We do this by enumerating all algorithms and modifying each algorithm to force it to terminate after $t(k)$ steps. Note that this enumeration will include all polynomial-time algorithms.

We denote the i-th encoding ensemble in this enumeration by $\mathcal{F}^i$ and the s-th member of that encoding ensemble by f_s^i. We let $\mathcal{U}$ denote the universal encoding ensemble given by

$$\mathcal{U}(\langle i, s\rangle, x) = f_s^i(x) \tag{9}$$

We remark that there exists a machine that computes $\mathcal{U}$ and runs in time $t(k)$. Now consider the relation R induced by $\mathcal{U}$ given by

$$R = \{(x, y) : y = \mathcal{U}(x, \bar{x})\} \subseteq \{0,1\}^* \times S \tag{10}$$

where $\bar{x}$ is the element of $\mathbb{Z}_p$ such that $\bar{x} \equiv x \bmod p$ (i.e. $(x, y) \in R$ if and only if $x = \langle i, s\rangle$ and $y = f_s^i(\bar{x})$). This relation is clearly evasive as for any x there exists at most one value of y such that $(x, y) \in R$. Again we consider the oracle D_R^σ such that

$$D_R^\sigma(y, \sigma(x)) = \begin{cases} x & \text{if } (y, \sigma(y)) \in R, \\ \perp & \text{otherwise.} \end{cases} \tag{11}$$

Now we may deduce the following two results in exactly the same way as before but using the evasive relation R (which is slightly different to the evasive group relation we used before).

Lemma 2. *If M^{σ,D_R^σ} is a generic algorithm that make a polynomial number of queries to any oracle then*

$$Pr[x \leftarrow M^{\sigma,D_R^\sigma}(\sigma(x))]$$

is negligible, where the probability is taken over the uniform choice of encoding function σ and the coins of M.

Now we replace the random encoding function σ with the label describing the encoding function, $\langle i,s\rangle$. It is important to replace σ with $\langle i,s\rangle$ exactly. Since σ is an oracle that is available to both the attacker and the oracle D_R we must make sure that both the attacker and the oracle have access to a legitimate copy of $\langle i,s\rangle$. It is easiest to think of $\langle i,s\rangle$ as a system parameter.

Lemma 3. *There exists a Turing machine M that runs in time polynomial in k such that*

$$Pr[x \leftarrow M^{D_R^{\langle i,s\rangle}}(f_s^i(x), \langle i,s\rangle)] = 1 \enspace .$$

Proof. M queries $D_R^{\langle i,s\rangle}$ with the input $(\langle i,s\rangle, f_s^i(x))$ and then outputs the output of the oracle. This can be done in polynomial time since we know f_s^i is a polynomial time encoding function. □

5 Signature Schemes

The results in this paper have been phrased in terms of an oracle problem that is provably hard in the generic group model. Some readers might dislike the use of a very powerful oracle that only outputs useful information in a very small number of cases. We have chosen to exhibit the results in the more general sense of a problem but it should also be noted that the above results could have been phrased in terms of a signature or encryption scheme. Here the oracle is replaced by access to a signing oracle or a decryption oracle, which seems much more natural.

5.1 A Signature Scheme that Runs in Super-Polynomial Time

Suppose $(\mathcal{S},\mathcal{V})$ is a signature scheme secure against adaptive chosen message attacks in the generic group model. We can modify the scheme so that it is still secure in the generic group model but insecure when any encoding ensemble is used instead of a random encoding function. Let $\mathcal{S}_1$ be the signing function given by

$$\mathcal{S}_1^\sigma(m,sk) = \begin{cases} sk||\mathcal{S}^\sigma(m,sk) & \text{if } (m,\sigma(m)) \in R, \\ \mathcal{S}^\sigma(m,sk) & \text{if } (m,\sigma(m)) \notin R. \end{cases} \tag{12}$$

where m is the message to be signed, sk is the secret key and R is the relation given in equation 10. The corresponding verifying algorithm, $\mathcal{V}_1$ is given by

$$\mathcal{V}_1^\sigma(m,s,pk) = \begin{cases} \mathcal{V}^\sigma(m,s',pk) & \text{if } (m,\sigma(m)) \in R \text{ and } s = x||s' \\ & \text{(where } x \text{ is the same length as } sk\text{)}, \\ \mathcal{V}^\sigma(m,s,pk) & \text{if } (m,\sigma(m)) \notin R. \end{cases} \tag{13}$$

where m is the message, s is the signature and pk is the public key. The signature scheme $(\mathcal{S}_1, \mathcal{V}_1)$ is still secure against adaptive chosen message attacks in the generic group model.

However we have already shown that once we replace the random encoding function σ with an encoding function f_s drawn at random from an encoding ensemble $\mathcal{F}^i$ then we can find a message $m = \langle i, s\rangle$ for which $(m, f_s^i(m)) \in R$. Hence we completely recover the secret key if we query the signing oracle with m. So the scheme is insecure for any concrete instantiation of the encoding function, i.e. the scheme is insecure in practice.

Unfortunately we are not quite finished: at the moment both the signing and verifying algorithms run in time $t(k) = O(k^{\log k})$. This is because both algorithms need to check a relation in R and then only way to check if $(\langle i', s'\rangle, y) \in R$ is to check if $f_{s'}^{i'}(\langle i', s'\rangle) = y$, which may take super-polynomial time.

5.2 Running the Scheme in Polynomial Time

We will use the CS-proof techniques of Micali [5] to run this scheme in polynomial time. Unlike [3] we cannot use guaranteed CS-proofs as we are unable to easily construct independent random functions[1], so we will instead use the notion of a *cryptographic CS-proof*. For this we require that all parties have access to a common random string r. Micali [5] shows that there exists polynomial-time algorithms Pro and Ver such that

- if $(x, \sigma(x)) \in R$ then Pro that computes a proof π to this fact,
- if $(x, \sigma(x)) \in R$ and π is a proof to this fact then Ver verifies this proof,
- if $(x, \sigma(x)) \notin R$ then a polynomial-time adversary produces a proof π' that Ver accepts for only an exponentially small number of random strings r.

From the details of [5] we see that it is reasonable to assume that the last fact goes even further: for any random string r it is computationally infeasible for a

[1] Of course, we could allow all parties to have access to a random oracle and then use the construction given in [3]. This would then allow us to prove that, in the random oracle model, there exists a scheme that is secure in the generic group model but insecure in any practical situation. Alternatively we could construct a scheme that is secure in the combined random oracle/generic group model but insecure in the standard model (i.e. when all random functions are replaced with functions drawn from the relevant ensembles). Whilst this technique is used successfully in Schnorr and Jakobsson [7] we feel that, in this particular situation, this is too much like passing the buck!

polynomial-time adversary to find a group element x and a proof π' such that $(x, \sigma(x)) \notin R$ but VER accepts the proof.

So we may now define a new signature scheme $(\mathcal{S}_2, \mathcal{V}_2)$ that is still secure against adaptive chosen ciphertext attacks in the generic group model. Note that any message m may be written as $x||\pi$ where x is a group element, hence we may define the signing function on a message m to be:

$$\mathcal{S}_2^\sigma(m, sk) = \begin{cases} sk||\mathcal{S}^\sigma(m, sk) & \text{if } \textsc{Ver} \text{ verifies the proof } \pi \text{ that} \\ & (x, \sigma(x)) \in R, \\ \mathcal{S}^\sigma(m, sk) & \text{otherwise.} \end{cases} \tag{14}$$

where sk is the secret key. The corresponding verifying function for a message m and a proposed signature s is given by:

$$\mathcal{V}_2^\sigma(m, s, pk) = \begin{cases} \mathcal{V}^\sigma(m, s', pk) & \text{if } \textsc{Ver} \text{ verifies the proof } \pi \text{ that} \\ & (x, \sigma(x)) \in R \text{ and } s = x||s' \text{ (where} \\ & x \text{ is the same length as } sk\text{),} \\ \mathcal{V}^\sigma(m, s, pk) & \text{otherwise.} \end{cases} \tag{15}$$

This scheme is secure in the generic group model because it is computationally infeasible to guess x such that $(x, \sigma(x)) \in R$ and it is also computationally infeasible to produce a proof π that will fool the signing oracle into believing that $(x, \sigma(x)) \in R$. Furthermore, since VER runs in polynomial-time, both the signing and verifying functions run in polynomial-time.

However when the random encoding function is replaced by the encoding function f_s^i then an attacker could submit the message

$$m = \langle i, s \rangle \;||\; \textsc{Pro}(x, r) \tag{16}$$

to the signing oracle and the signing oracle will return the secret key. So the scheme is insecure for any practical instantiation of the encoding function.

6 Conclusion

We have shown that the generic group model suffers from the same weaknesses as the random oracle model, namely, that a problem can be shown to be hard in the generic group model but is easy when the random function is changed to any specific function or set of functions. This shows that the generic group model is not a perfect way to represent an algorithm that attacks a problem defined on a group but doesn't take advantage of any of the specific group structure.

We have also adapted this to show that there are cryptographic schemes that are secure in the generic group model that are insecure whenever a specific encoding function is used. Heuristically this means that security proofs that rely on the generic group model should be viewed with the same caution as security proofs that rely on the random oracle model.

Acknowledgements

The author would like to thank the Information Security Group of Royal Holloway and, in particular, Kenny Paterson and Christine Swart for helpful discussions on this subject.

References

1. M. Bellare and P. Rogaway. 'Random Oracles are Practical: A Paradigm for Designing Effecient Protocols.' *Proceedings of the First ACM Conference on Computer and Communications Security*, 1993.
2. D. Brown. 'Generic Groups, Collision Resistance, and ECDSA.' Available from `http://eprint.iacr.org/`, 2002.
3. R. Canetti, O. Goldreich and S. Halevi. 'The Random Oracle Methodology, Revisited.' *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, 1998.
4. M. Fischlin. 'A Note on Security Proofs in the Generic Model.' *Advances in Cryptology - Asiacrypt 2000*, 2000.
5. S. Micali. 'CS proofs.' *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, 1994.
6. C. Schnorr. 'Efficient Signature Generation for Smart Cards.' *Journal of Cryptology*, Vol 4, 1991.
7. C. Schnorr and M. Jakobsson. 'Security of Signed El-Gamel Encryption.' *Advances in Cryptology - Asiacrypt 2000*, 2000.
8. V. Shoup. 'Lower Bounds for Discrete Logarithms and Related Problems.' *Theory and Application of Cryptographic Techniques*, 1997.
9. N. Smart. 'The Exact Security of ECIES in the Generic Group Model.' *Cryptography and Coding*, 2001.