Spreading Alerts Quietly:

New Insights from Theory and Practice

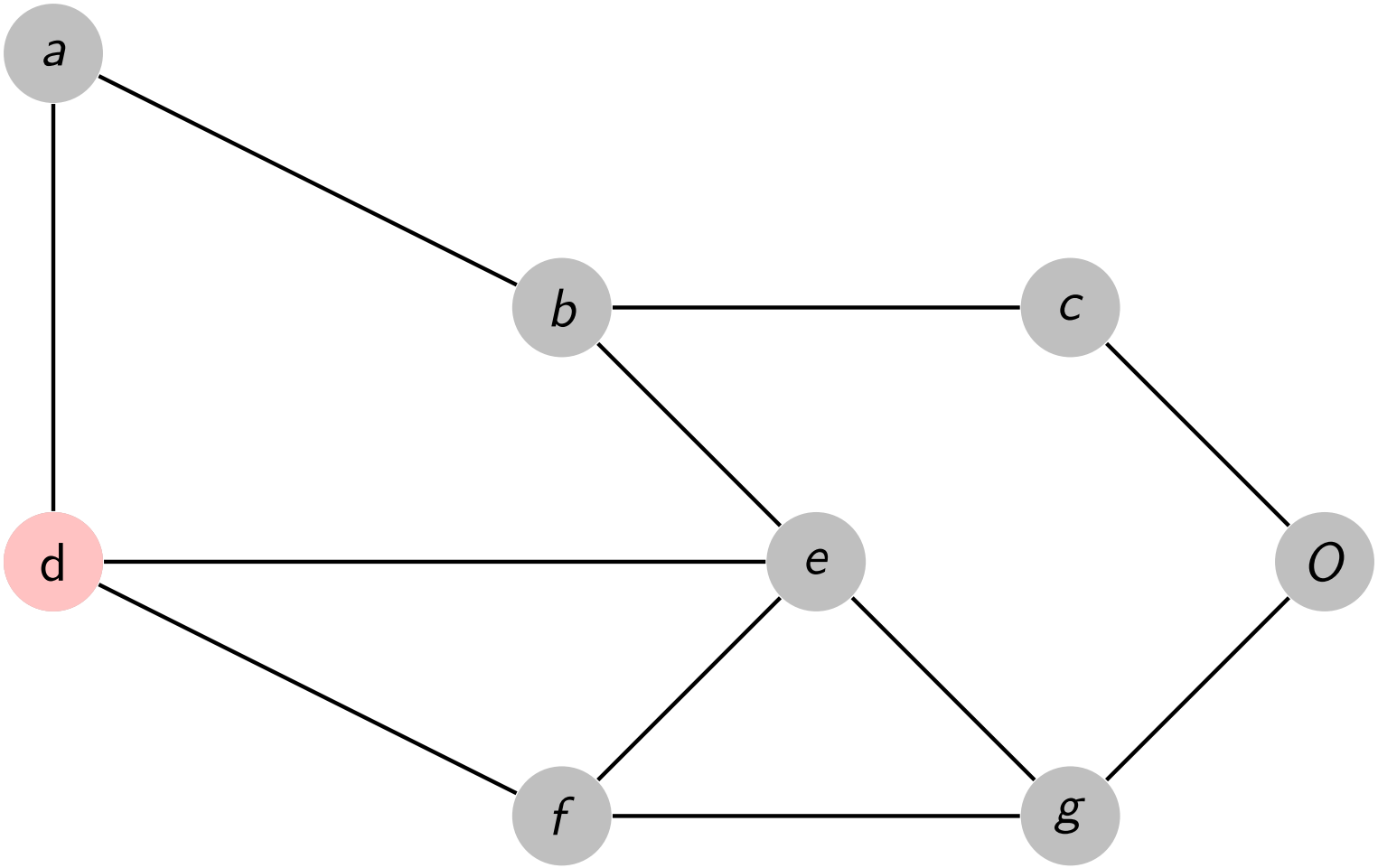
O. Blazy C. Chevalier

ARES, August 2018

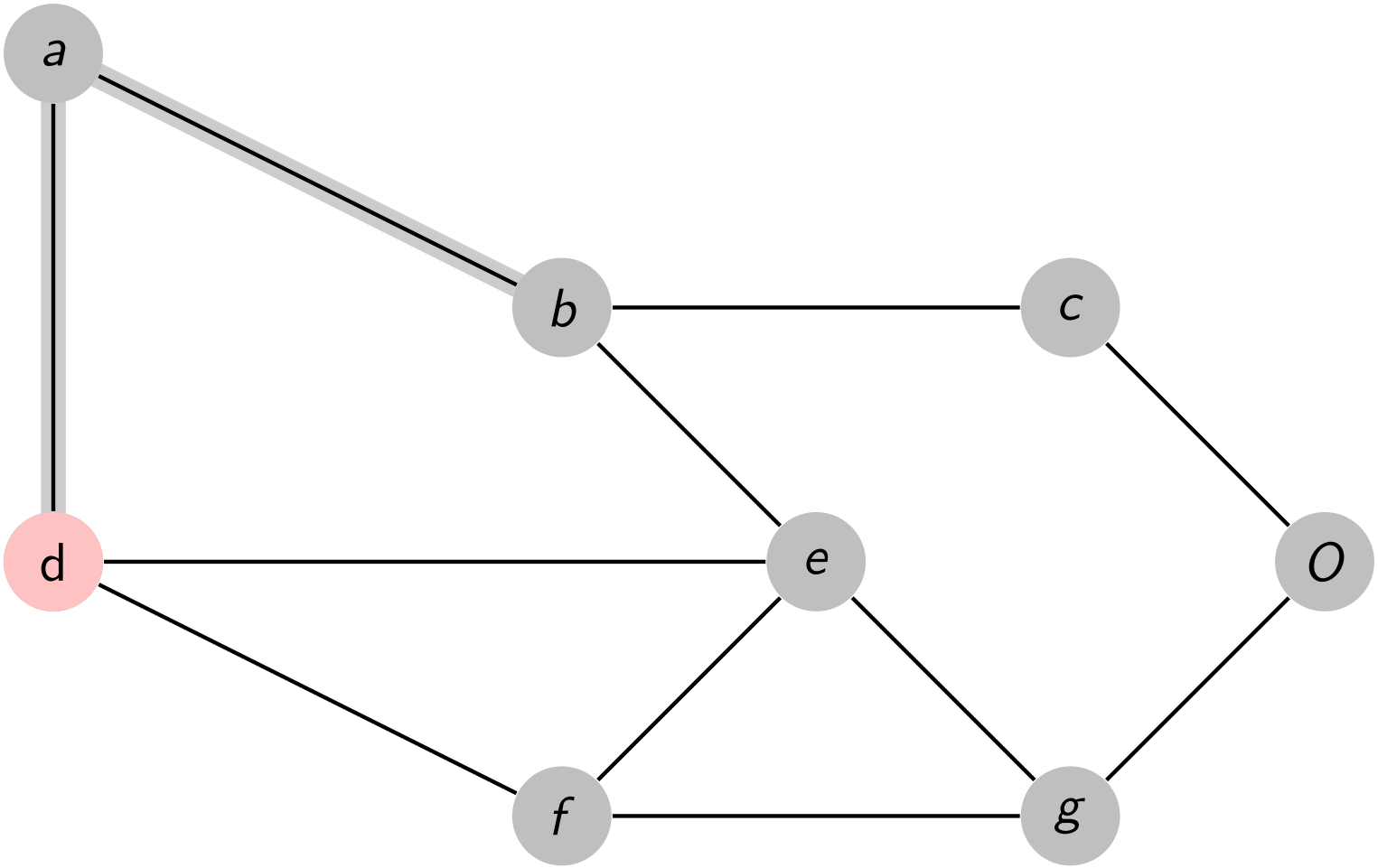
INSTITUT DE RECHERCHE xlim

UNIVERSITÉ PARIS II PANTHÉON-ASSAS

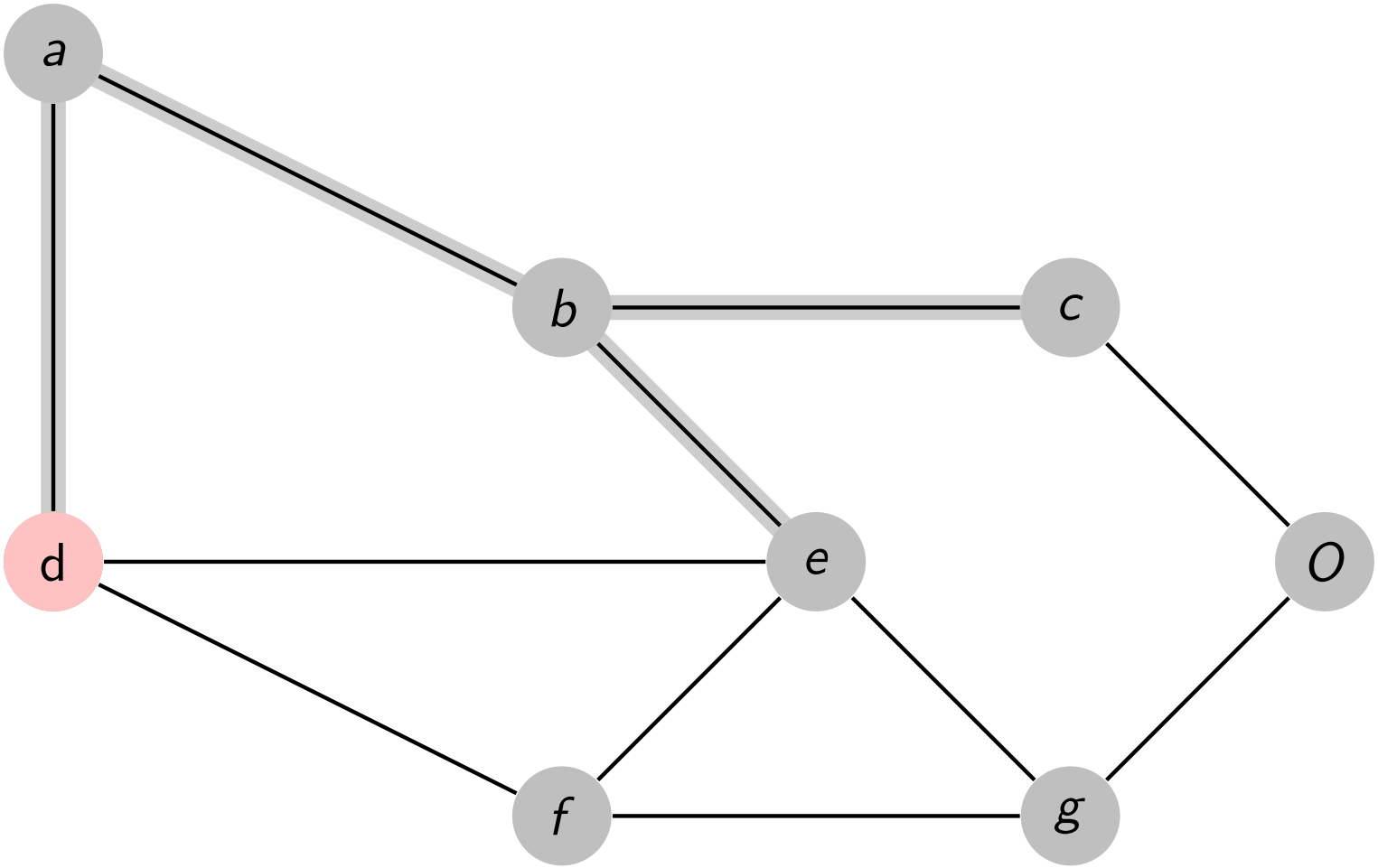
Spreading Alert



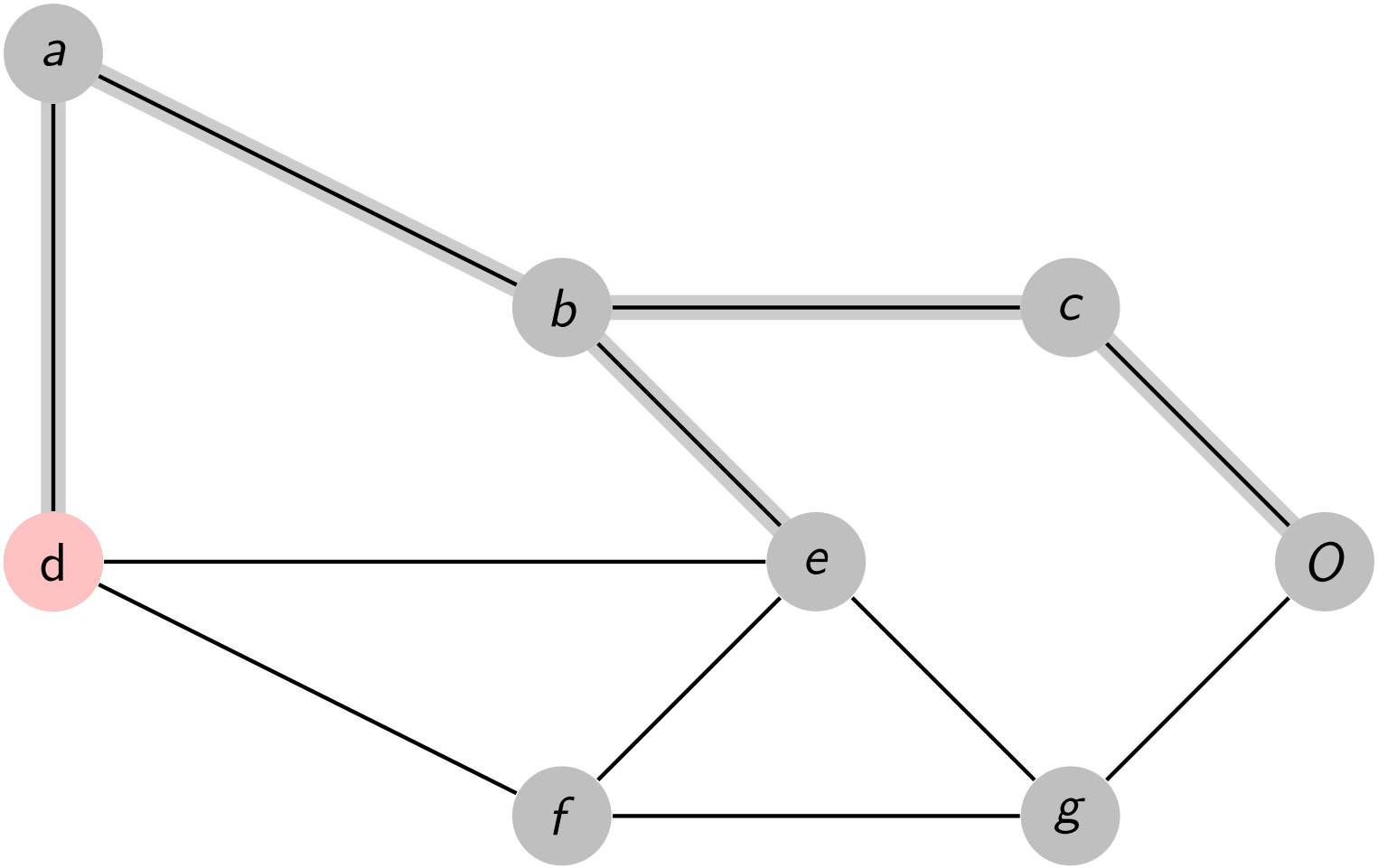
Spreading Alert



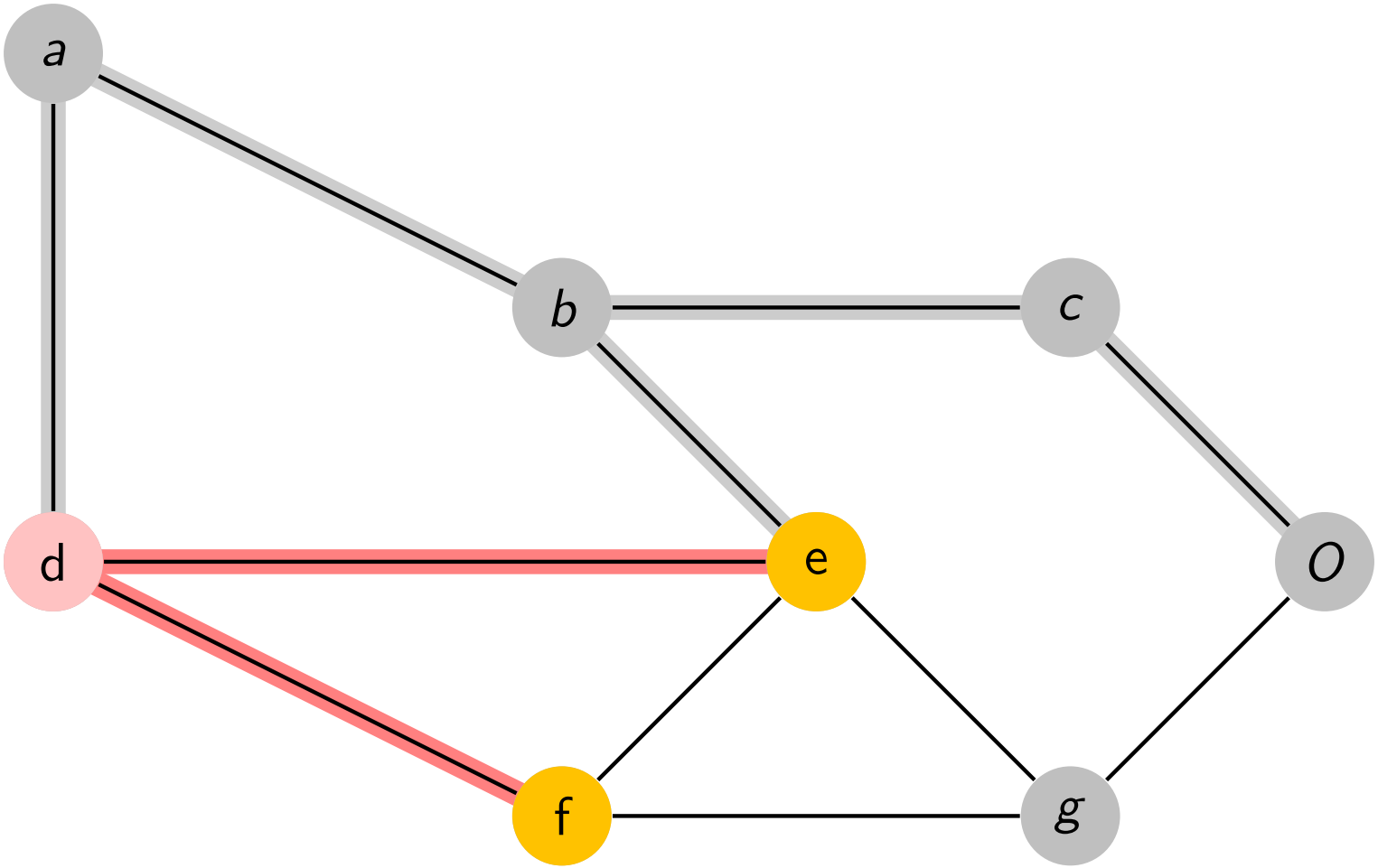
Spreading Alert



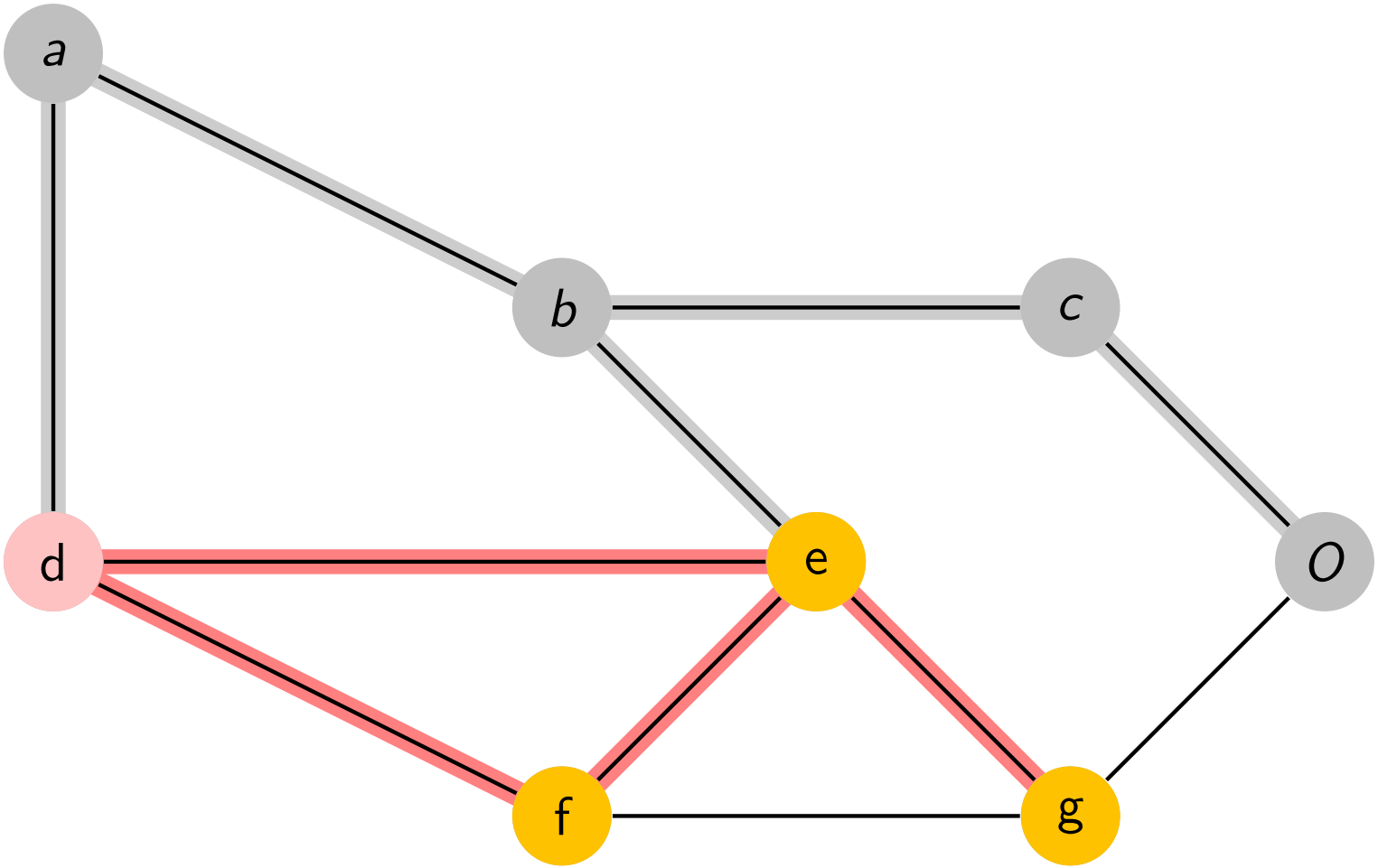
Spreading Alert



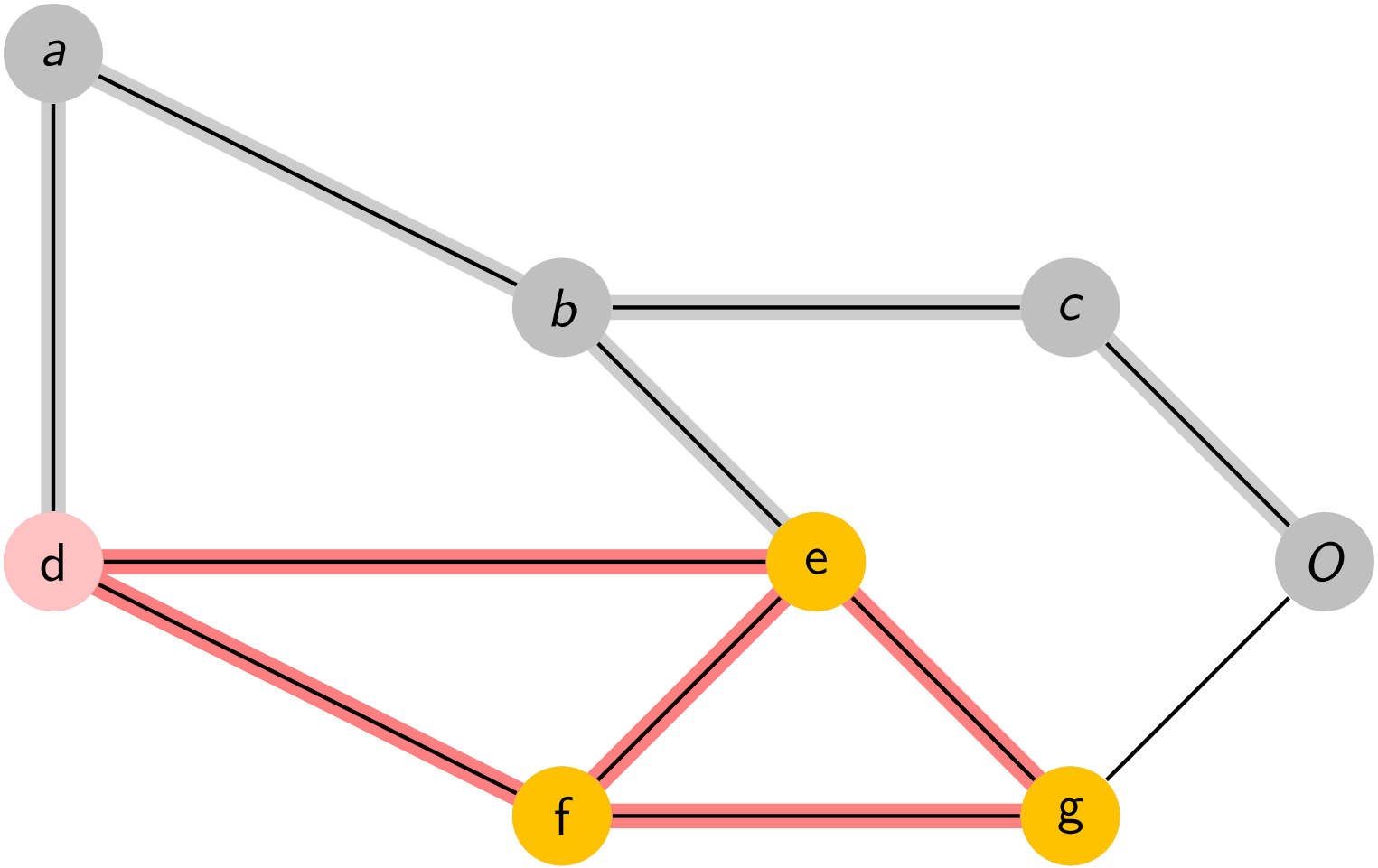
Spreading Alert



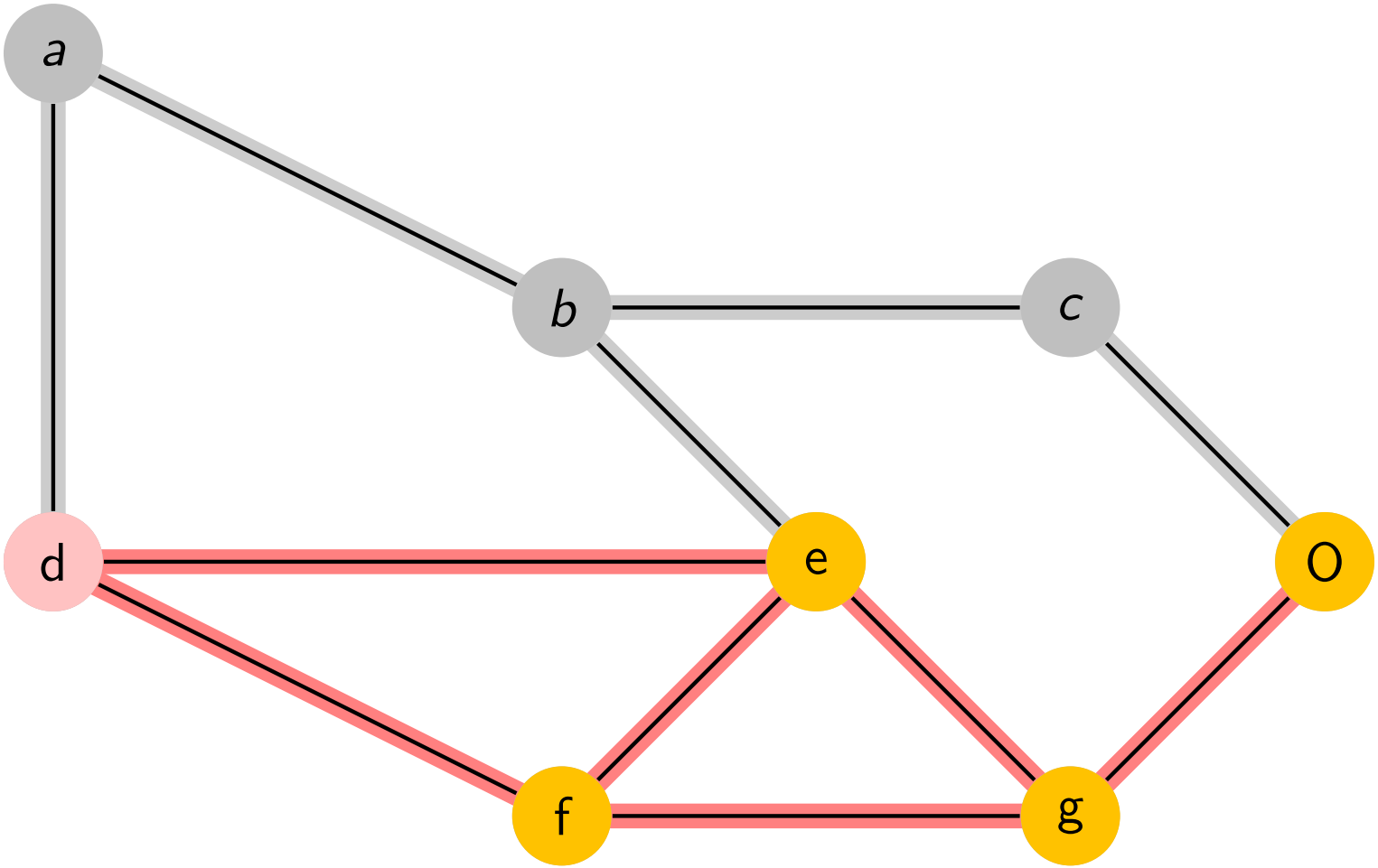
Spreading Alert



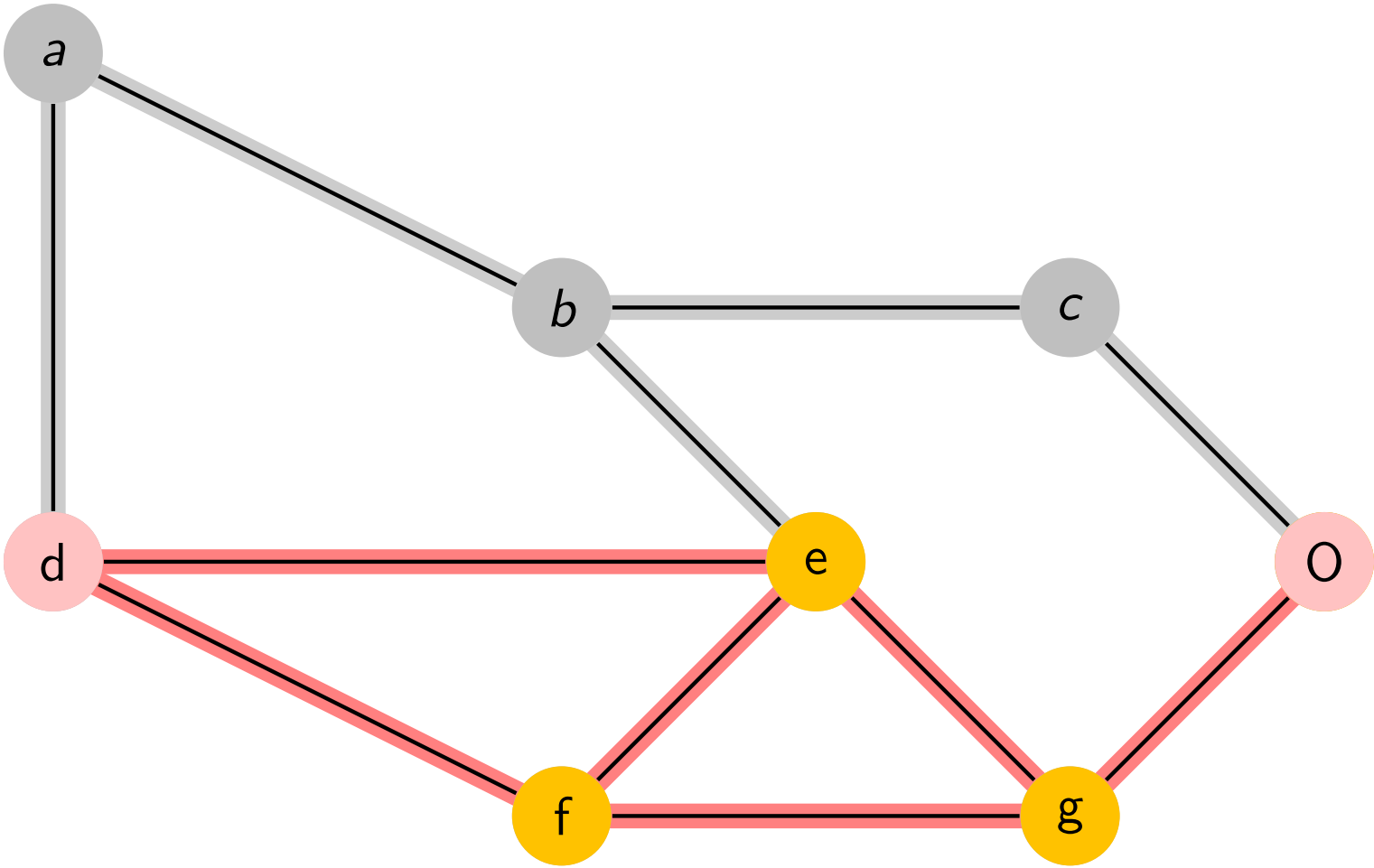
Spreading Alert



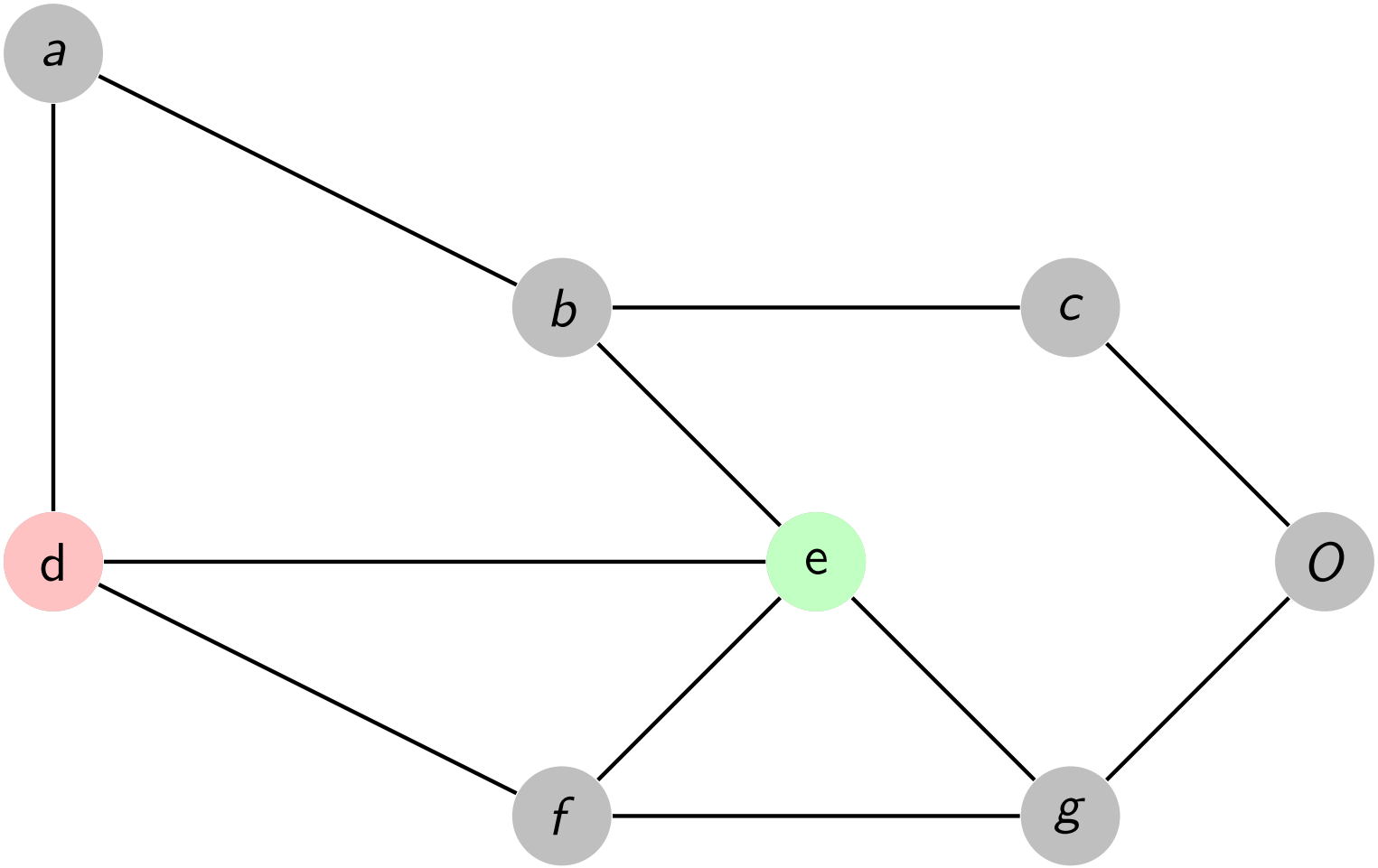
Spreading Alert



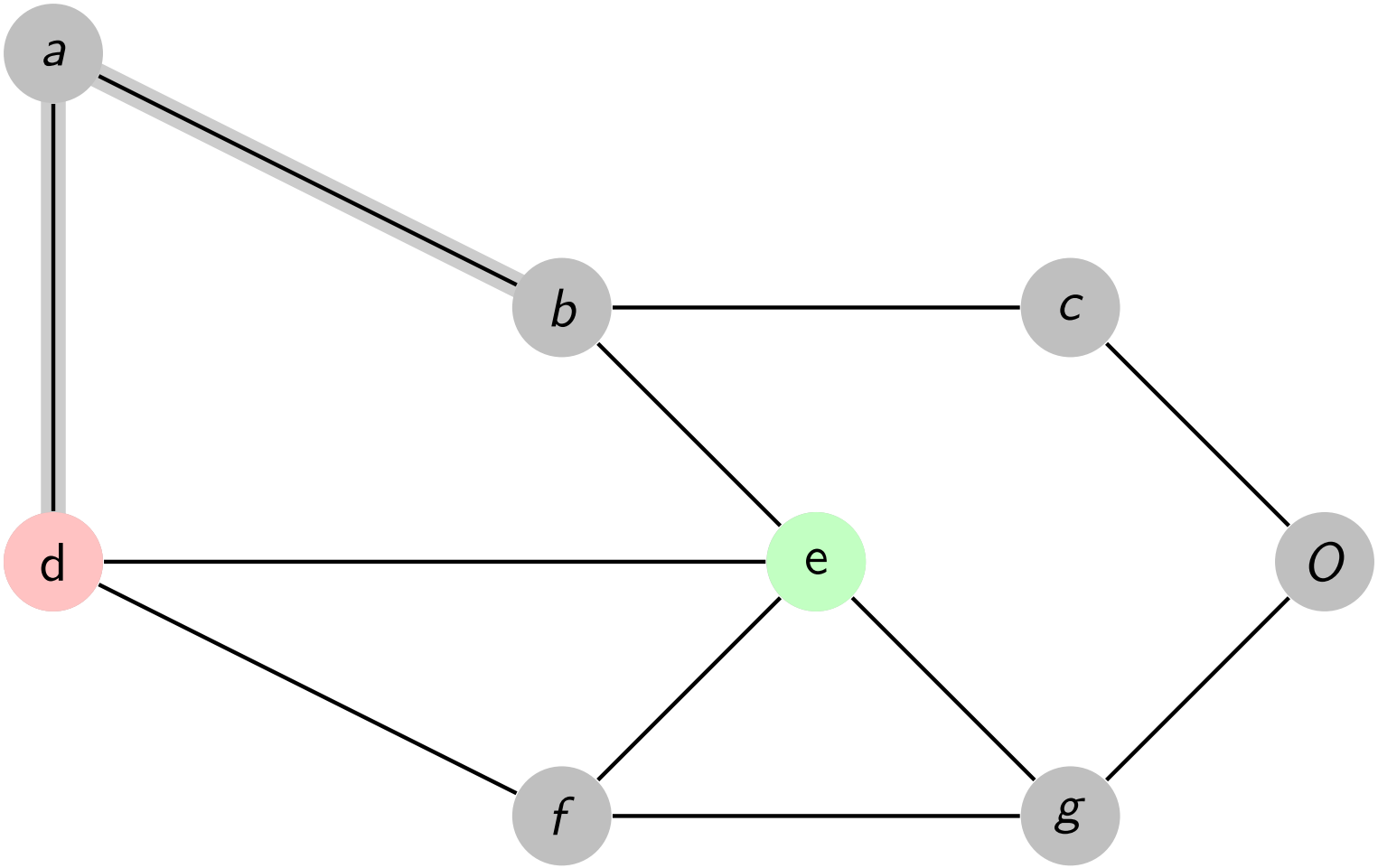
Spreading Alert



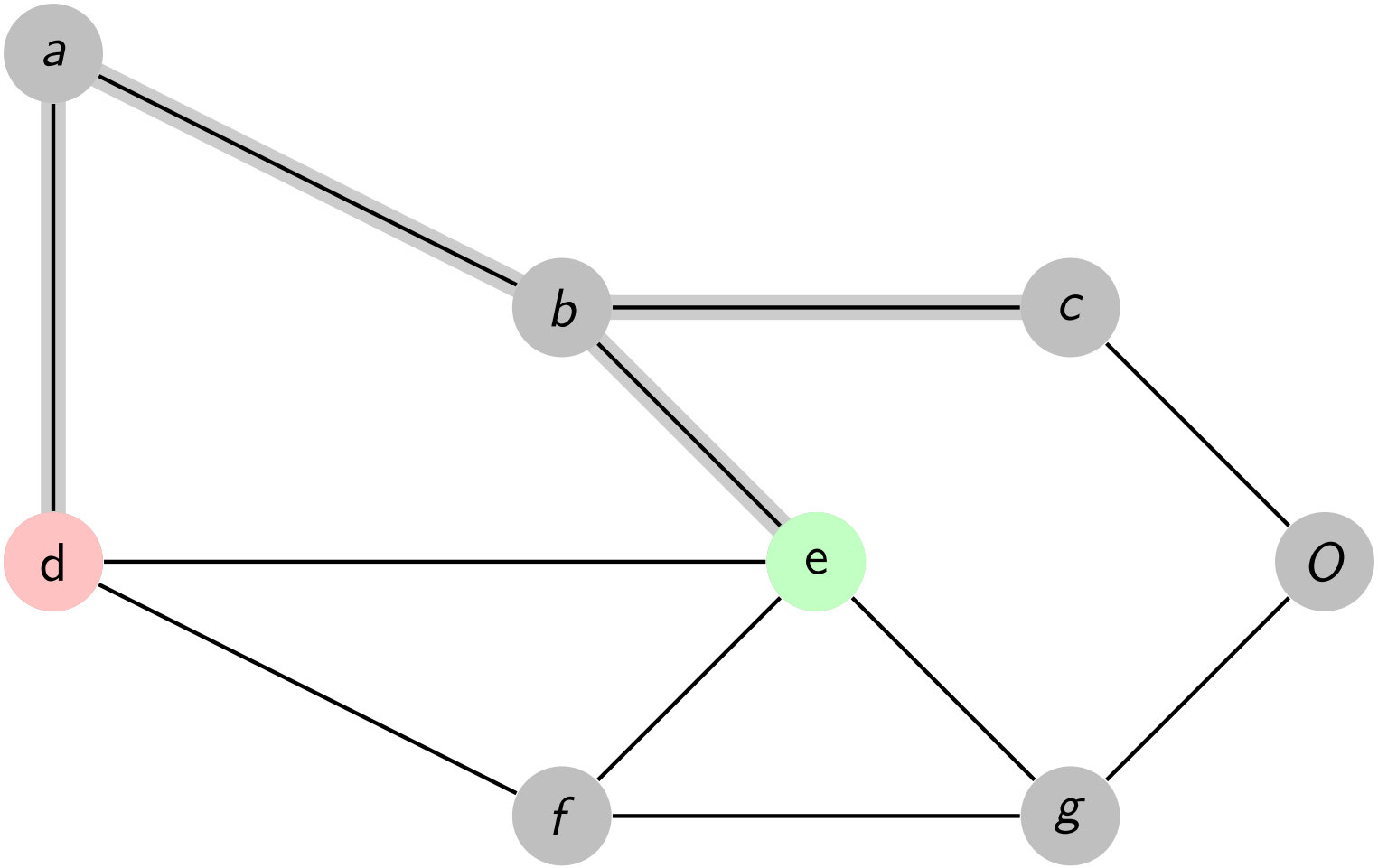
Spreading Alert



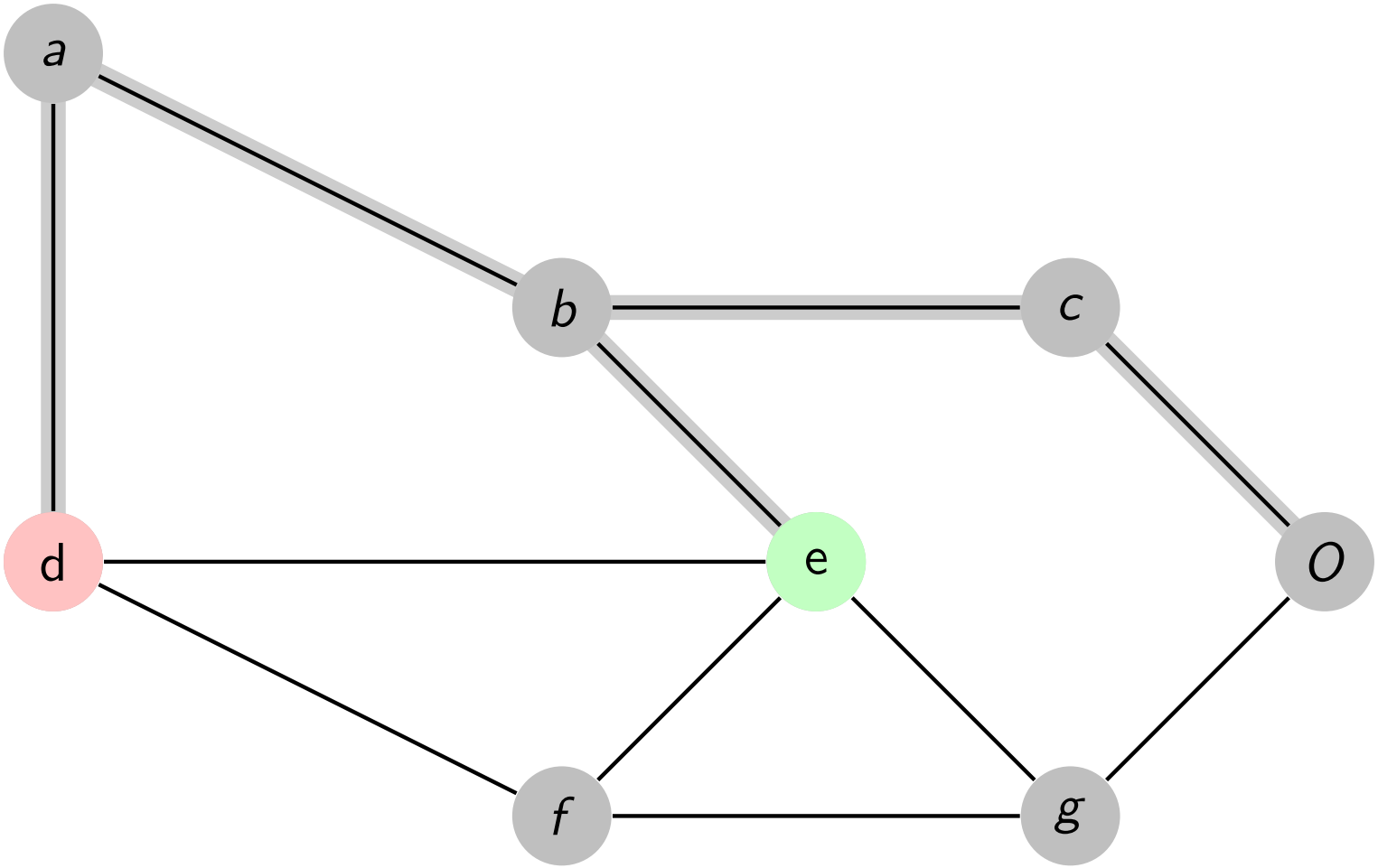
Spreading Alert



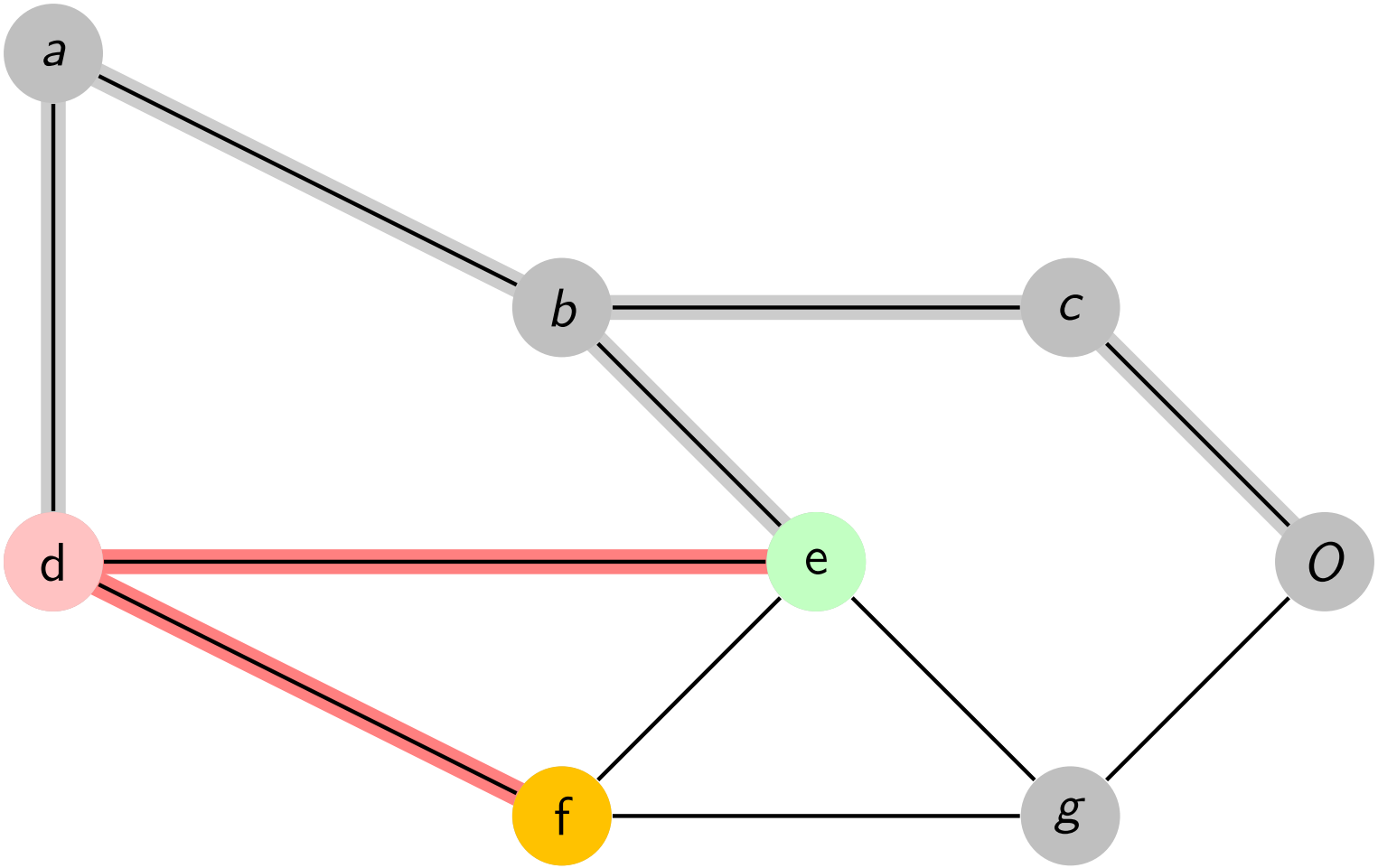
Spreading Alert



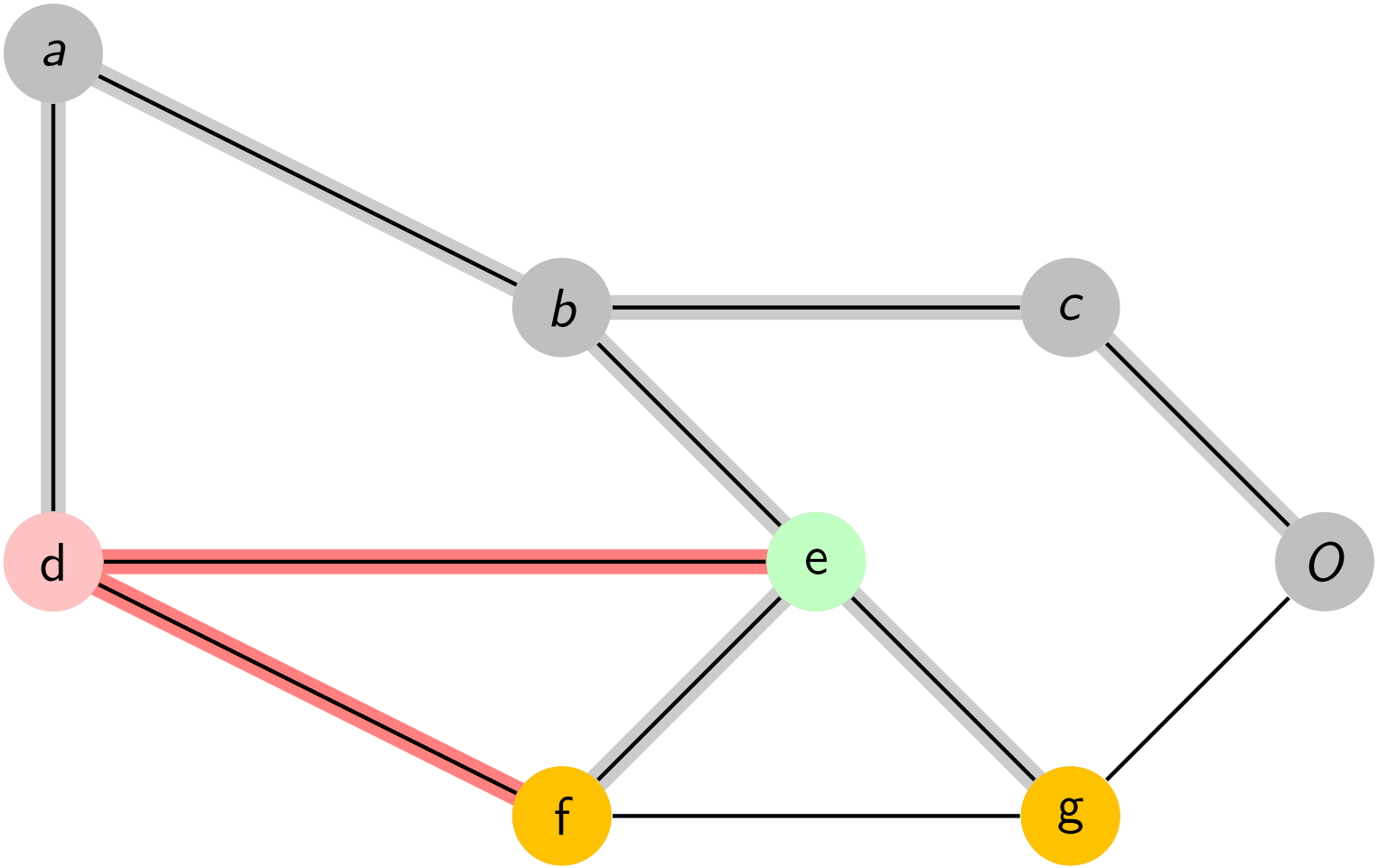
Spreading Alert



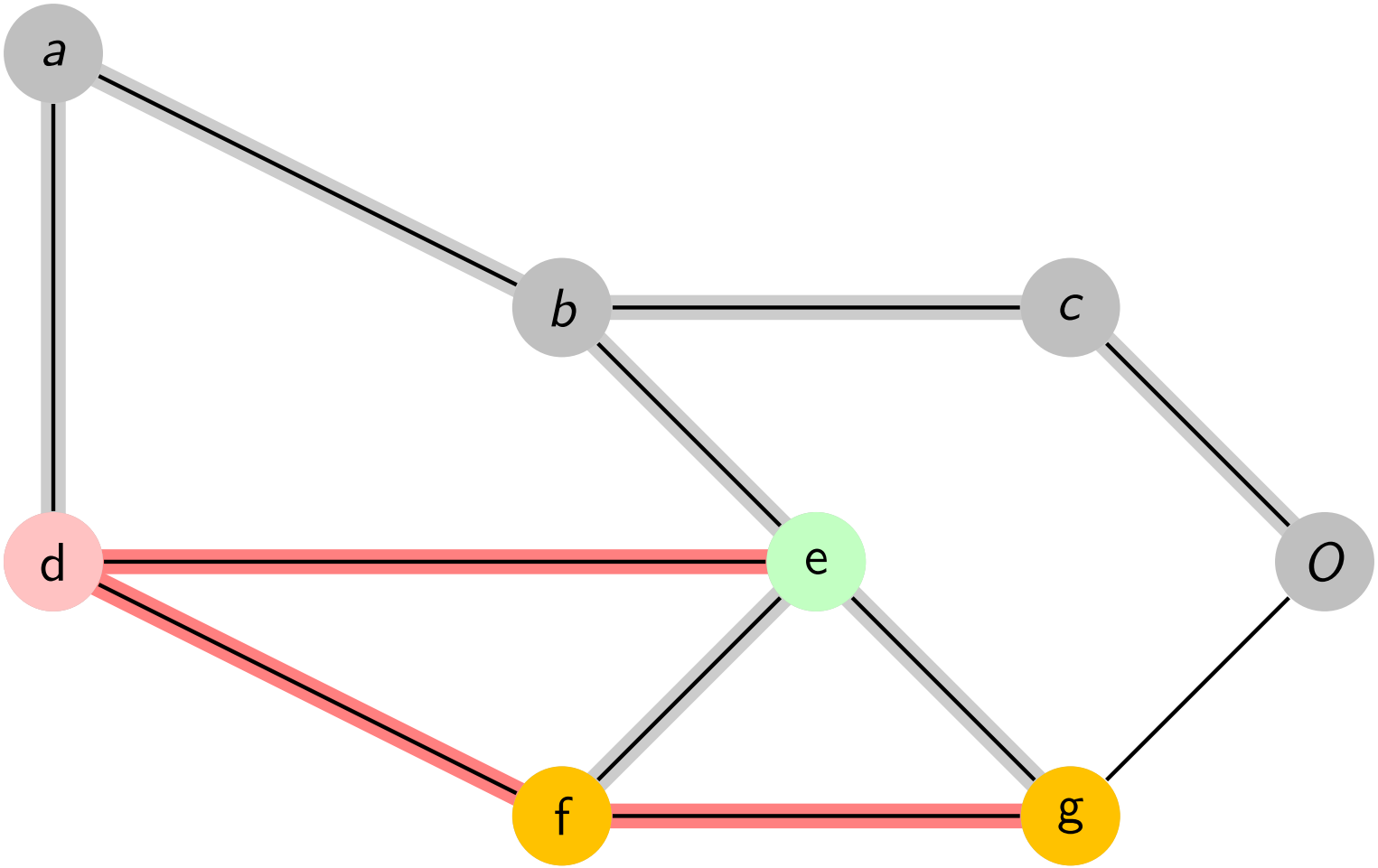
Spreading Alert



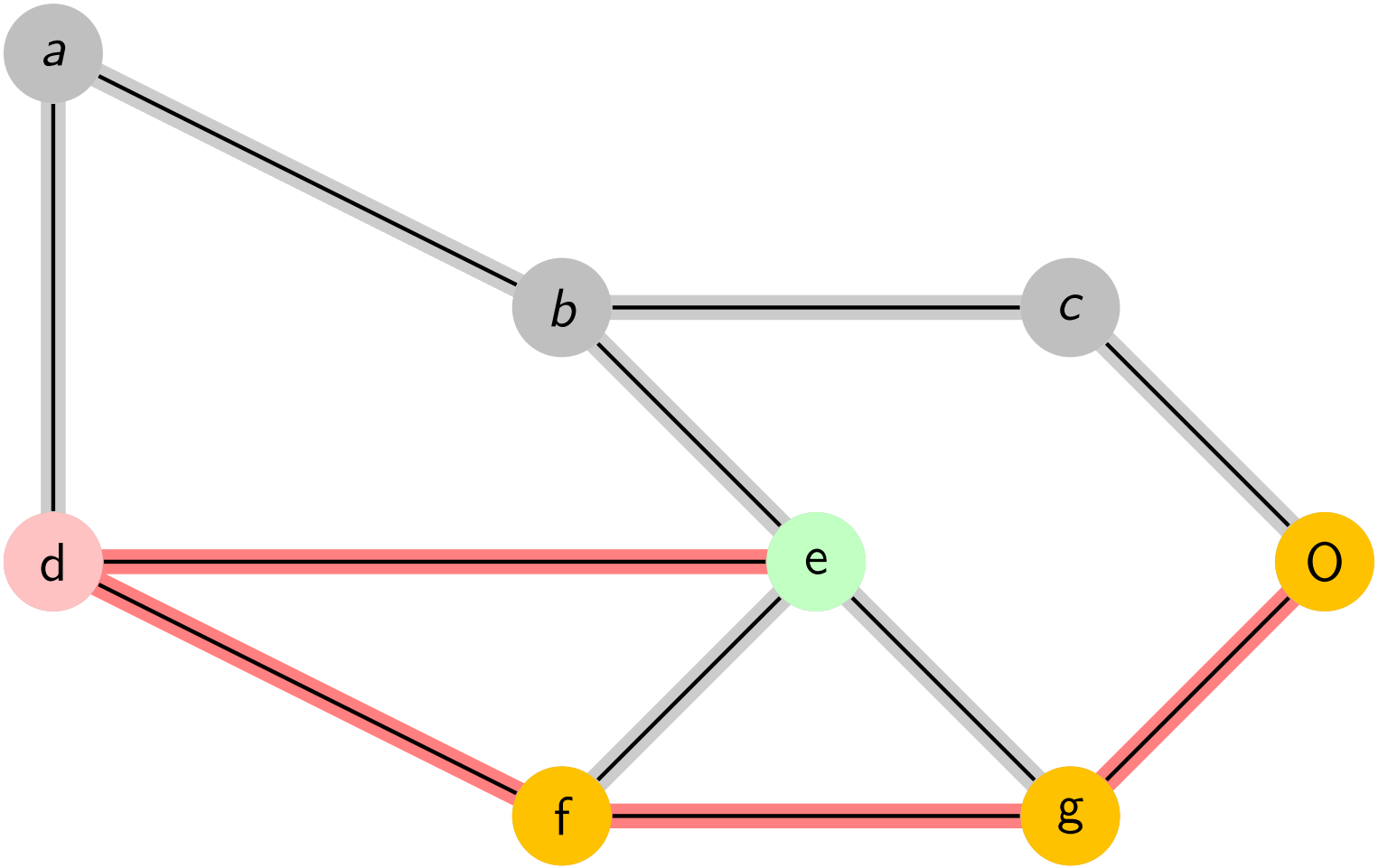
Spreading Alert



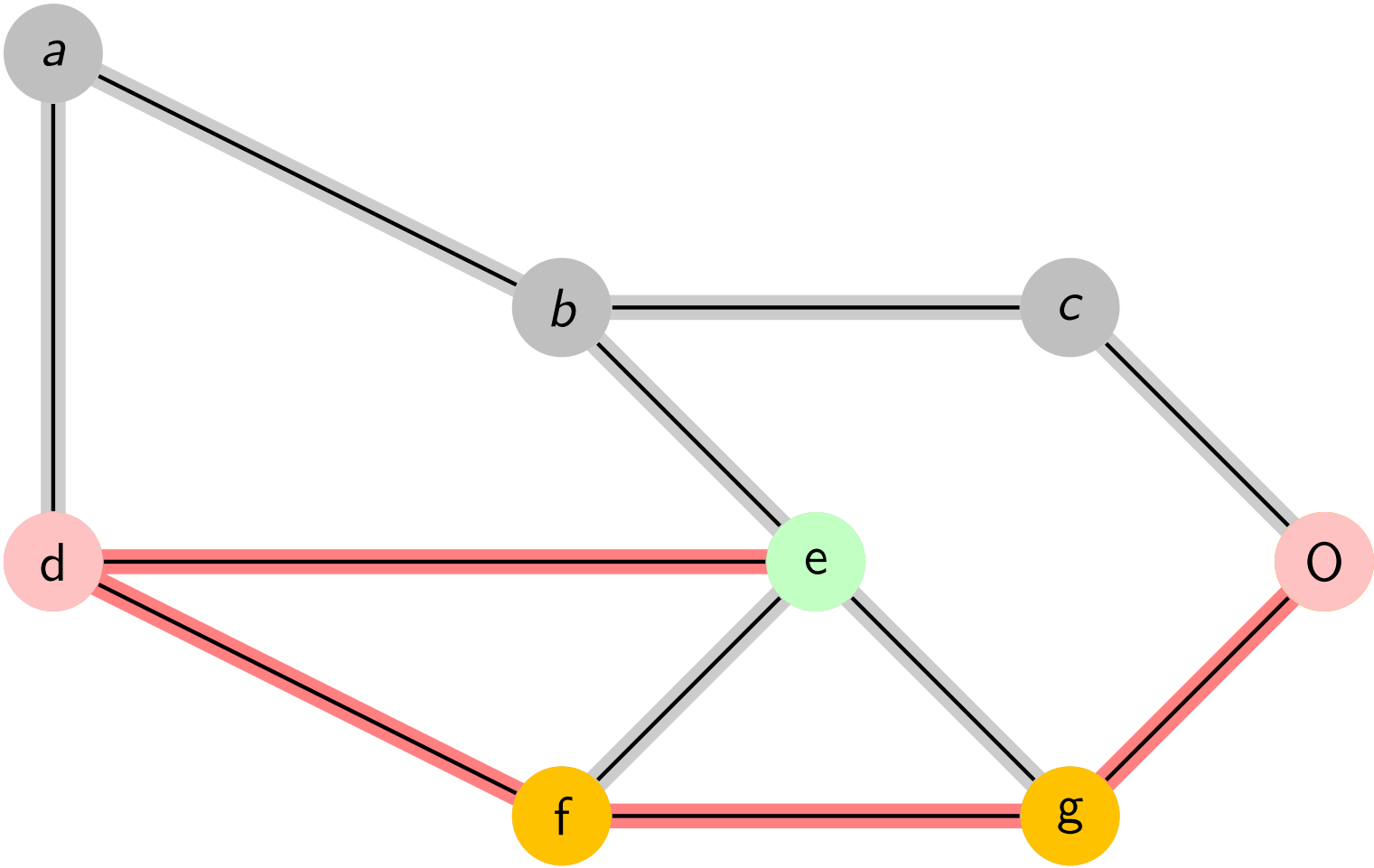
Spreading Alert



Spreading Alert



Spreading Alert



Concerns

Spreading Alert

- No other attack than denial of service
- O should be active if there is at least one alert

Quietly

Intermediate block should not learn if there is an alert

- Initial Model from [ADYGP05]

- Initial Model from [ADYGP05]
- Too loose, it can be unconditionally secure

- Initial Model from [ADYGP05]
- Too loose, it can be unconditionally secure
- Adversary should be allowed to see signals!

- Initial Model from [ADYGP05]
- Too loose, it can be unconditionally secure
- Adversary should be allowed to see signals!
- (Strongly) Connected Graph of non faulty nodes

Blind Coupon Mechanism

- BCMGen(1^λ) outputs (pk, sk):

Blind Coupon Mechanism

- BCMGen(1^λ) outputs (pk, sk):
  - pk for the set of valid coupons V_{sk} included into the universe set U_{sk}.

Blind Coupon Mechanism

- BCMGen(1^λ) outputs (pk, sk):
  - pk for the set of valid coupons V_{sk} included into the universe set U_{sk}.
  - sk is a secret key for D_{sk} and S_{sk}, such that $D_{\mathsf{sk}} \oplus S_{\mathsf{sk}} = V_{\mathsf{sk}}$;

Blind Coupon Mechanism

- BCMGen(1^λ) outputs (pk, sk):
  - pk for the set of valid coupons V_{sk} included into the universe set U_{sk}.
  - sk is a secret key for D_{sk} and S_{sk}, such that $D_{\mathsf{sk}} \oplus S_{\mathsf{sk}} = V_{\mathsf{sk}}$;
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}(1^\lambda)$ outputs (d, s) with $d \xleftarrow{\$} D_{\mathsf{sk}}$ and $s \xleftarrow{\$} S_{\mathsf{sk}}$;

Blind Coupon Mechanism

- $\mathsf{BCMGen}(1^\lambda)$ outputs $(\mathsf{pk}, \mathsf{sk})$:
  - pk for the set of valid coupons V_{sk} included into the universe set U_{sk}.
  - sk is a secret key for D_{sk} and S_{sk}, such that $D_{\mathsf{sk}} \oplus S_{\mathsf{sk}} = V_{\mathsf{sk}}$;
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}(1^\lambda)$ outputs (d, s) with $d \xleftarrow{\$} D_{\mathsf{sk}}$ and $s \xleftarrow{\$} S_{\mathsf{sk}}$;
- $\mathsf{BCMVerify}_{\mathsf{pk}}(c)$ returns 1 iff c is valid;

Blind Coupon Mechanism

- $\mathsf{BCMGen}(1^\lambda)$ outputs $(\mathsf{pk}, \mathsf{sk})$:
  - pk for the set of valid coupons V_{sk} included into the universe set U_{sk}.
  - sk is a secret key for D_{sk} and S_{sk}, such that $D_{\mathsf{sk}} \oplus S_{\mathsf{sk}} = V_{\mathsf{sk}}$;
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}(1^\lambda)$ outputs (d, s) with $d \xleftarrow{\$} D_{\mathsf{sk}}$ and $s \xleftarrow{\$} S_{\mathsf{sk}}$;
- $\mathsf{BCMVerify}_{\mathsf{pk}}(c)$ returns 1 iff c is valid;
- $\mathsf{BCMCombine}_{\mathsf{pk}}(c, y)$ returns a coupon z such that $z \in D_{\mathsf{sk}}$ if $c \in D_{\mathsf{sk}}$ and $y \in D_{\mathsf{sk}}$, and $z \in S_{\mathsf{sk}}$ otherwise;

Blind Coupon Mechanism

- $\mathsf{BCMGen}(1^\lambda)$ outputs $(\mathsf{pk}, \mathsf{sk})$:
  - pk for the set of valid coupons V_{sk} included into the universe set U_{sk}.
  - sk is a secret key for D_{sk} and S_{sk}, such that $D_{\mathsf{sk}} \oplus S_{\mathsf{sk}} = V_{\mathsf{sk}}$;
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}(1^\lambda)$ outputs (d, s) with $d \xleftarrow{\$} D_{\mathsf{sk}}$ and $s \xleftarrow{\$} S_{\mathsf{sk}}$;
- $\mathsf{BCMVerify}_{\mathsf{pk}}(c)$ returns 1 iff c is valid;
- $\mathsf{BCMCombine}_{\mathsf{pk}}(c, y)$ returns a coupon z such that $z \in D_{\mathsf{sk}}$ if $c \in D_{\mathsf{sk}}$ and $y \in D_{\mathsf{sk}}$, and $z \in S_{\mathsf{sk}}$ otherwise;
- $\mathsf{BCMDecode}_{\mathsf{sk}}(c)$ returns 1 if c is a signal coupon (belonging to S_{sk}) and 0 if it is a dummy coupon (belonging to D_{sk});

Indistinguishability

Given access to polynomially many valid dummy/signal coupons the adversary should not be able to distinguish the impact of a fresh valid coupon.

Indistinguishability

Given access to polynomially many valid dummy/signal coupons the adversary should not be able to distinguish the impact of a fresh valid coupon.

Unforgeability

Given access to many valid dummy coupons the adversary should not be able to create a valid signal coupon.

- BCMGen. $\mathsf{pk} = (p, \mathbb{G}, g)$. $x \xleftarrow{\$} \mathbb{Z}_p$, sets $h = g^x \in \mathbb{G}$. $\mathsf{sk} = (g, h, x)$.

- BCMGen. $\mathsf{pk} = (p, \mathbb{G}, g)$. $x \xleftarrow{\$} \mathbb{Z}_p$, sets $h = g^x \in \mathbb{G}$. $\mathsf{sk} = (g, h, x)$.
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}$ Sets $d \xleftarrow{\$} (g_1, g_2) \in \mathbb{G}^2$. $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $s = (g^r, h^r) \in \mathbb{G}^2$. It outputs d, s as dummy, signal coupon.

- BCMGen. $\mathsf{pk} = (p, \mathbb{G}, g)$. $x \xleftarrow{\$} \mathbb{Z}_p$, sets $h = g^x \in \mathbb{G}$. $\mathsf{sk} = (g, h, x)$.
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}$ Sets $d \xleftarrow{\$} (g_1, g_2) \in \mathbb{G}^2$. $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $s = (g^r, h^r) \in \mathbb{G}^2$. It outputs d, s as dummy, signal coupon.
- $\mathsf{BCMVerify}_{\mathsf{pk}}$. Given c, outputs 1 iff $c \in \mathbb{G}^2$ and $c \neq (1_{\mathbb{G}}, 1_{\mathbb{G}})$.

- BCMGen. $\mathsf{pk} = (p, \mathbb{G}, g)$. $x \xleftarrow{\$} \mathbb{Z}_p$, sets $h = g^x \in \mathbb{G}$. $\mathsf{sk} = (g, h, x)$.
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}$ Sets $d \xleftarrow{\$} (g_1, g_2) \in \mathbb{G}^2$. $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $s = (g^r, h^r) \in \mathbb{G}^2$. It outputs d, s as dummy, signal coupon.
- $\mathsf{BCMVerify}_{\mathsf{pk}}$. Given c, outputs 1 iff $c \in \mathbb{G}^2$ and $c \neq (1_{\mathbb{G}}, 1_{\mathbb{G}})$.
- $\mathsf{BCMCombine}_{\mathsf{pk}}$. Given two valid coupons c and y, outputs $z = c \cdot y$.

- BCMGen. $\mathsf{pk} = (p, \mathbb{G}, g)$. $x \xleftarrow{\$} \mathbb{Z}_p$, sets $h = g^x \in \mathbb{G}$. $\mathsf{sk} = (g, h, x)$.
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}$ Sets $d \xleftarrow{\$} (g_1, g_2) \in \mathbb{G}^2$. $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $s = (g^r, h^r) \in \mathbb{G}^2$. It outputs d, s as dummy, signal coupon.
- $\mathsf{BCMVerify}_{\mathsf{pk}}$. Given c, outputs 1 iff $c \in \mathbb{G}^2$ and $c \neq (1_{\mathbb{G}}, 1_{\mathbb{G}})$.
- $\mathsf{BCMCombine}_{\mathsf{pk}}$. Given two valid coupons c and y, outputs $z = c \cdot y$.
- $\mathsf{BCMDecode}_{\mathsf{sk}}$. Given a valid $c = (c_1, c_2)$, outputs 1 iff $c_2 = c_1^x$.

Combination (Absorbing)

Dummy-Dummy $\equiv$ Dummy

$(g_1, g_2) \cdot (f_1, f_2) \equiv (h_1, h_2) \xleftarrow{\$} \mathbb{G}^2$

Combination (Absorbing)

Dummy-Dummy $\equiv$ Dummy

$$(g_1, g_2) \cdot (f_1, f_2) \equiv (h_1, h_2) \xleftarrow{\$} \mathbb{G}^2$$

Dummy-Signal $\equiv$ Dummy

$$(g_1, g_2) \cdot (g^r, h^r) \equiv (h_1, h_2) \xleftarrow{\$} \mathbb{G}^2$$

Combination (Absorbing)

Dummy-Dummy $\equiv$ Dummy

$$(g_1, g_2) \cdot (f_1, f_2) \equiv (h_1, h_2) \xleftarrow{\$} \mathbb{G}^2$$

Dummy-Signal $\equiv$ Dummy

$$(g_1, g_2) \cdot (g^r, h^r) \equiv (h_1, h_2) \xleftarrow{\$} \mathbb{G}^2$$

Signal-Signal $\equiv$ Signal

$$(g^r, h^r) \cdot (g^s, h^s) \equiv (g^t, h^t) \xleftarrow{\$} \mathbb{G}^2$$

Indistinguishability

Under the DDH assumption

Indistinguishability

Under the DDH assumption

Unforgeability

Impossible to generate a Diffie Hellman tuple in an unknown basis.

Linearly Homomorphic Signature

- Introduced in [LPJY13]

Linearly Homomorphic Signature

- Introduced in [LPJY13]
- Allows to have signatures that can be combined in a session (tag)

Linearly Homomorphic Signature

- Introduced in [LPJY13]
- Allows to have signatures that can be combined in a session (tag)
- Need to be tweaked to be efficient

Linearly Homomorphic Signature

- KeyGen(λ): Outputs a key pair vk, sk and a tag space $\mathcal{T}$.

Linearly Homomorphic Signature

- KeyGen(λ): Outputs a key pair vk, sk and a tag space $\mathcal{T}$.
- Sign(sk, τ, M): Outputs a signature $\sigma \in \mathbb{G}_1^i \times \mathbb{G}_2^j$.

Linearly Homomorphic Signature

- KeyGen(λ): Outputs a key pair vk, sk and a tag space $\mathcal{T}$.
- Sign(sk, τ, M): Outputs a signature $\sigma \in \mathbb{G}_1^i \times \mathbb{G}_2^j$.
- SignDerive(vk, τ, $(\lambda_j, M_j, \sigma(M_j))_j$): Returns σ on $M = \prod_j M_j^{\lambda_j}$.

Linearly Homomorphic Signature

- KeyGen(λ): Outputs a key pair vk, sk and a tag space $\mathcal{T}$.
- Sign(sk, τ, M): Outputs a signature $\sigma \in \mathbb{G}_1^i \times \mathbb{G}_2^j$.
- SignDerive(vk, τ, $(\lambda_j, M_j, \sigma(M_j))_j$): Returns σ on $M = \prod_j M_j^{\lambda_j}$.
- Verify(vk, τ, M, σ): Outputs 1 iff the signature is valid.

Efficient Asymmetric Version

- KeyGen(λ):

Efficient Asymmetric Version

- KeyGen(λ):
  1. Picks $\alpha_h \xleftarrow{\$} \mathbb{Z}_p$, sets $h_2 = g_2^{\alpha_h}$

Efficient Asymmetric Version

- KeyGen(λ):
  1. Picks $\alpha_h \xleftarrow{\$} \mathbb{Z}_p$, sets $h_2 = g_2^{\alpha_h}$
  2. For $i \in \{0,1\}$, picks $\xi_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$, sets $\mathsf{hp}_i = g_2^{\xi_i} h_2^{\delta_i}$

  $\mathsf{sk} = (\xi_i, \delta_i)$, $h_2 \in \mathbb{Z}_p^{2\times 2} \times \mathbb{G}_2$, and $\mathsf{vk} = (h_2, \mathsf{hp}_i) \in \mathbb{G}_2 \times \mathbb{G}_2^2$

Efficient Asymmetric Version

- KeyGen(λ):
  1. Picks $\alpha_h \xleftarrow{\$} \mathbb{Z}_p$, sets $h_2 = g_2^{\alpha_h}$
  2. For $i \in \{0,1\}$, picks $\xi_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$, sets $\mathsf{hp}_i = g_2^{\xi_i} h_2^{\delta_i}$

  $\mathsf{sk} = (\xi_i, \delta_i)$, $h_2 \in \mathbb{Z}_p^{2\times 2} \times \mathbb{G}_2$, and $\mathsf{vk} = (h_2, \mathsf{hp}_i) \in \mathbb{G}_2 \times \mathbb{G}_2^2$
- Sign(sk, ϵ, M): $\sigma = (z,u) = (M_0, M_1) \odot (-\xi_i, -\delta_i)$:

$$z = M_0^{-\xi_0} M_1^{-\xi_1}, u = M_0^{-\delta_0} M_1^{-\delta_1}$$

Efficient Asymmetric Version

- KeyGen(λ):
  1. Picks $\alpha_h \xleftarrow{\$} \mathbb{Z}_p$, sets $h_2 = g_2^{\alpha_h}$
  2. For $i \in \{0,1\}$, picks $\xi_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$, sets $\mathsf{hp}_i = g_2^{\xi_i} h_2^{\delta_i}$

  $\mathsf{sk} = (\xi_i, \delta_i)$, $h_2 \in \mathbb{Z}_p^{2\times 2} \times \mathbb{G}_2$, and $\mathsf{vk} = (h_2, \mathsf{hp}_i) \in \mathbb{G}_2 \times \mathbb{G}_2^2$
- Sign(sk, ϵ, M): $\sigma = (z,u) = (M_0, M_1) \odot (-\xi_i, -\delta_i)$:

$$z = M_0^{-\xi_0} M_1^{-\xi_1}, u = M_0^{-\delta_0} M_1^{-\delta_1}$$

- SignDerive($\mathsf{vk}, \epsilon, (\lambda_j, M_j, \sigma(M_j) = (z_j, u_j))$: Simply computes

$$z = \prod z_j^{\lambda_j}, u = \prod u_j^{\lambda_j}$$

Efficient Asymmetric Version

- KeyGen(λ):
  1. Picks $\alpha_h \xleftarrow{\$} \mathbb{Z}_p$, sets $h_2 = g_2^{\alpha_h}$
  2. For $i \in \{0,1\}$, picks $\xi_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p$, sets $\mathsf{hp}_i = g_2^{\xi_i} h_2^{\delta_i}$

  $\mathsf{sk} = (\xi_i, \delta_i)$, $h_2 \in \mathbb{Z}_p^{2\times 2} \times \mathbb{G}_2$, and $\mathsf{vk} = (h_2, \mathsf{hp}_i) \in \mathbb{G}_2 \times \mathbb{G}_2^2$
- Sign(sk, ϵ, M): $\sigma = (z,u) = (M_0, M_1) \odot (-\xi_i, -\delta_i)$:

$$z = M_0^{-\xi_0} M_1^{-\xi_1}, u = M_0^{-\delta_0} M_1^{-\delta_1}$$

- SignDerive($\mathsf{vk}, \epsilon, (\lambda_j, M_j, \sigma(M_j) = (z_j, u_j))$: Simply computes

$$z = \prod z_j^{\lambda_j}, u = \prod u_j^{\lambda_j}$$

- Verify($\mathsf{vk}, \epsilon, M, \sigma$): checks whether

$$1_{\mathbb{G}_T} = e(z, g_2) e(u, h_2) e(M_0, \mathsf{hp}_0) e(M_1, \mathsf{hp}_1)$$

Combining

- BCMGen(1^λ): $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sign.KeyGen}()$, $\beta_h \xleftarrow{\$} \mathbb{Z}_p$, $g \xleftarrow{\$} \mathbb{G}_1$, $h = g^{\beta_h}$, sets $\mathsf{pk} = \mathsf{vk}, \mathsf{sk} = (\mathsf{sk}, g, h, \beta_h)$

Combining

- BCMGen(1^λ): $(\mathsf{vk}, \mathsf{sk}) \leftarrow$ Sign.KeyGen(), $\beta_h \xleftarrow{\$} \mathbb{Z}_p$, $g \xleftarrow{\$} \mathbb{G}_1$, $h = g^{\beta_h}$, sets $\mathsf{pk} = \mathsf{vk}, \mathsf{sk} = (\mathsf{sk}, g, h, \beta_h)$
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}(1^\lambda)$: Picks $\rho_i, \rho'_i \xleftarrow{\$} \mathbb{Z}_p^*$, outputs a dummy $((g^{\rho_i}, h^{\rho_i}), \mathsf{Sign}(\mathsf{sk}, (g^{\rho_i}, h^{\rho_i})))$, and a signal $((g^{\rho_i}, h^{\rho'_i}), \mathsf{Sign}(\mathsf{sk}, (g^{\rho_i}, h^{\rho'_i})))$.

Combining

- BCMGen(1^λ): $(\mathsf{vk}, \mathsf{sk}) \leftarrow$ Sign.KeyGen(), $\beta_h \xleftarrow{\$} \mathbb{Z}_p$, $g \xleftarrow{\$} \mathbb{G}_1$, $h = g^{\beta_h}$, sets $\mathsf{pk} = \mathsf{vk}, \mathsf{sk} = (\mathsf{sk}, g, h, \beta_h)$
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}(1^\lambda)$: Picks $\rho_i, \rho_i' \xleftarrow{\$} \mathbb{Z}_p^*$, outputs a dummy $((g^{\rho_i}, h^{\rho_i}), \mathsf{Sign}(\mathsf{sk}, (g^{\rho_i}, h^{\rho_i})))$, and a signal $((g^{\rho_i}, h^{\rho_i'}), \mathsf{Sign}(\mathsf{sk}, (g^{\rho_i}, h^{\rho_i'})))$.
- $\mathsf{BCMVerify}_{\mathsf{pk}}(c)$: Outputs Sign.Verify$(\mathsf{vk}, \epsilon, c_1, c_2)$.

Combining

- BCMGen(1^λ): $(\mathsf{vk}, \mathsf{sk}) \leftarrow$ Sign.KeyGen(), $\beta_h \xleftarrow{\$} \mathbb{Z}_p$, $g \xleftarrow{\$} \mathbb{G}_1$, $h = g^{\beta_h}$, sets $\mathsf{pk} = \mathsf{vk}, \mathsf{sk} = (\mathsf{sk}, g, h, \beta_h)$
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}(1^\lambda)$: Picks $\rho_i, \rho_i' \xleftarrow{\$} \mathbb{Z}_p^*$, outputs a dummy $((g^{\rho_i}, h^{\rho_i}), \mathsf{Sign}(\mathsf{sk}, (g^{\rho_i}, h^{\rho_i})))$, and a signal $((g^{\rho_i}, h^{\rho_i'}), \mathsf{Sign}(\mathsf{sk}, (g^{\rho_i}, h^{\rho_i'})))$.
- $\mathsf{BCMVerify}_{\mathsf{pk}}(c)$: Outputs $\mathsf{Sign.Verify}(\mathsf{vk}, \epsilon, c_1, c_2)$.
- $\mathsf{BCMCombine}_{\mathsf{pk}}(c_u, c_R)$: Picks $\lambda, \mu \xleftarrow{\$} \mathbb{Z}_p$ and outputs $\mathsf{SignDerive}(\mathsf{vk}, (\lambda, c_u, \mu c_R))$.

Combining

- BCMGen(1^λ): $(\mathsf{vk}, \mathsf{sk}) \leftarrow$ Sign.KeyGen(), $\beta_h \xleftarrow{\$} \mathbb{Z}_p$, $g \xleftarrow{\$} \mathbb{G}_1$, $h = g^{\beta_h}$, sets $\mathsf{pk} = \mathsf{vk}, \mathsf{sk} = (\mathsf{sk}, g, h, \beta_h)$
- $\mathsf{BCMCouponGen}_{\mathsf{pk},\mathsf{sk}}(1^\lambda)$: Picks $\rho_i, \rho_i' \xleftarrow{\$} \mathbb{Z}_p^*$, outputs a dummy $((g^{\rho_i}, h^{\rho_i}), \mathsf{Sign}(\mathsf{sk}, (g^{\rho_i}, h^{\rho_i})))$, and a signal $((g^{\rho_i}, h^{\rho_i'}), \mathsf{Sign}(\mathsf{sk}, (g^{\rho_i}, h^{\rho_i'})))$.
- $\mathsf{BCMVerify}_{\mathsf{pk}}(c)$: Outputs $\mathsf{Sign.Verify}(\mathsf{vk}, \epsilon, c_1, c_2)$.
- $\mathsf{BCMCombine}_{\mathsf{pk}}(c_u, c_R)$: Picks $\lambda, \mu \xleftarrow{\$} \mathbb{Z}_p$ and outputs $\mathsf{SignDerive}(\mathsf{vk}, (\lambda, c_u, \mu c_R))$.
- $\mathsf{BCMDecode}_{\mathsf{sk}}(\mathsf{sk}, c)$: Using β_h, checks whether $c_2 / c_1^{\beta_h} \neq 1_{\mathbb{G}_1}$.

Conclusion

	Base Order	Memory Used	Signal Size	Combine Cost	Verification
[ADGPY05]	$n = pqz$	2 $\mathbb{G}_n$	1 $\mathbb{G}_n$	1 M	1 P
Us (Trivial)	Prime	4 $\mathbb{G}$	2 $\mathbb{G}$	2 M	1 E
Us (Full)	Prime	6 $\mathbb{G}_1$ + 4 $\mathbb{G}_2$	3 $\mathbb{G}_2$ + 2 $\mathbb{G}_1$	6 P + 8 E + 4 M	6 P

✓ Better security (Unconditional and/or Stronger Model)

Conclusion

	Base Order	Memory Used	Signal Size	Combine Cost	Verification
[ADGPY05]	$n = pqz$	2 $\mathbb{G}_n$	1 $\mathbb{G}_n$	1 M	1 P
Us (Trivial)	Prime	4 $\mathbb{G}$	2 $\mathbb{G}$	2 M	1 E
Us (Full)	Prime	6 $\mathbb{G}_1$ + 4 $\mathbb{G}_2$	3 $\mathbb{G}_2$ + 2 $\mathbb{G}_1$	6 P + 8 E + 4 M	6 P

- ✓ Better security (Unconditional and/or Stronger Model)
- ✓ Efficient (66% for Unconditional, 133% for Stronger)

Conclusion

	Base Order	Memory Used	Signal Size	Combine Cost	Verification
[ADGPY05]	$n = pqz$	2 $\mathbb{G}_n$	1 $\mathbb{G}_n$	1 M	1 P
Us (Trivial)	Prime	4 $\mathbb{G}$	2 $\mathbb{G}$	2 M	1 E
Us (Full)	Prime	6 $\mathbb{G}_1$ + 4 $\mathbb{G}_2$	3 $\mathbb{G}_2$ + 2 $\mathbb{G}_1$	6 P + 8 E + 4 M	6 P

- ✓ Better security (Unconditional and/or Stronger Model)
- ✓ Efficient (66% for Unconditional, 133% for Stronger)
- ✓ Modular

Conclusion

	Base Order	Memory Used	Signal Size	Combine Cost	Verification
[ADGPY05]	$n = pqz$	2 $\mathbb{G}_n$	1 $\mathbb{G}_n$	1 M	1 P
Us (Trivial)	Prime	4 $\mathbb{G}$	2 $\mathbb{G}$	2 M	1 E
Us (Full)	Prime	6 $\mathbb{G}_1$ + 4 $\mathbb{G}_2$	3 $\mathbb{G}_2$ + 2 $\mathbb{G}_1$	6 P + 8 E + 4 M	6 P

- ✓ Better security (Unconditional and/or Stronger Model)
- ✓ Efficient (66% for Unconditional, 133% for Stronger)
- ✓ Modular
- ✓ New instantiation of a LHSPS

Conclusion

	Base Order	Memory Used	Signal Size	Combine Cost	Verification
[ADGPY05]	$n = pqz$	2 $\mathbb{G}_n$	1 $\mathbb{G}_n$	1 M	1 P
Us (Trivial)	Prime	4 $\mathbb{G}$	2 $\mathbb{G}$	2 M	1 E
Us (Full)	Prime	6 $\mathbb{G}_1$ + 4 $\mathbb{G}_2$	3 $\mathbb{G}_2$ + 2 $\mathbb{G}_1$	6 P + 8 E + 4 M	6 P

- ✓ Better security (Unconditional and/or Stronger Model)
- ✓ Efficient (66% for Unconditional, 133% for Stronger)
- ✓ Modular
- ✓ New instantiation of a LHSPS
- ⇛ A post-quantum solution?